Enhanced Autonomous Driving: PrediNet20 with AHLR for Improved Performance

Chuanji Xu[1] and Jian Qu[2*]

[1,2]Faculty of Engineering and Technology, Panyapiwat Institute of Management, Nonthaburi, Thailand
E-mail: 6572100065@stu.pim.ac.th, jianqu@pim.ac.th

Received: October 20, 2023/ Revised: February 2, 2024/ Accepted: February 14, 2024

***Abstract*—In the continually evolving field of autonomous driving, enhancing model prediction accuracy and addressing noisy data remain pivotal challenges. This study introduces PrediNet20, a customized end-to-end Convolutional Neural Network (CNN) designed for the Donkey Car platform. PrediNet20 aims to alleviate the limitations of current deep learning models by improving accuracy in predicting throttle and steering angles, crucial components in autonomous driving systems. At the core of this enhancement is the introduction of AHLR, a novel adaptive loss function that enhances model training and generalization. It dynamically adjusts the loss based on the prediction error, facilitating a smooth transition from quadratic to linear loss. Coupled with the application of L1 regularization, it aids in reducing overfitting, potentially enhancing the model's resistance to data noise and outliers. Preliminary experiments using real driving data indicate that compared to existing models, PrediNet20 demonstrates approximately a 33.3% improvement in convergence speed, a 37.5% improvement in stability, a 10% improvement in robustness, and a 50% improvement in generalization. PrediNet20 offers higher accuracy and faster convergence, marking a significant step forward in the development of more reliable autonomous driving systems.**

***Index Terms*—End-to-end, Autonomous Driving, Deep Learning, Predinet20, AHLR**

I. Introduction

End-to-end-based autonomous driving has been recognized as the future of automated driving [1], promising to revolutionize the way we perceive and engage with transportation. Rapid advances in artificial intelligence and machine learning have revolutionized and laid the foundation for safer and more trustworthy autonomous driving systems. However, improving model prediction accuracy, handling noisy data, and avoiding training overfitting remain very critical issues. Thus, there is a need to continuously develop and improve the relevant models and algorithms that would produce a more rationalized autonomous driving system [2]. Among these models and algorithms within the autonomous driving system, improvements in deep learning neural networks and loss functions usually yield better results in terms of decision rationalization for autonomous vehicles.

Deep learning, as a pivotal subset of machine learning, has emerged as a crucial tool in the development of autonomous driving systems, particularly excelling in handling high-dimensional data and intricate nonlinear relationships [3], [4]. In the realm of autonomous driving, Convolutional Neural Networks (CNNs) have been extensively employed for tasks such as target detection, semantic segmentation, and end-to-end driving. Despite achieving certain successes, these models still exhibit limitations in performance and reliability when predicting throttle and steering wheel angle values from images generated in autonomous driving experiments.

Current models often grapple with overfitting issues and struggle to handle anomalies and noise in training data, potentially compromising the model's generalization ability and performance in real-world scenarios. To address these challenges, this study introduces a novel CNN-based deep learning network architecture, namely PrediNet20, tailored specifically for the Donkey Car autonomous driving platform. This model leverages meticulously designed convolutional layers combined with LeakyReLUs and filter regularization to efficiently capture spatial features of input images [5], [6]. PrediNet20 aims to overcome the performance and reliability limitations of current models, with a particular focus on enhancing the robustness of the model against anomalies and noise to improve performance in practical scenarios.

Adaptive loss function, AHLR is one of the most important key innovations in our proposed PrediNet20. This function dynamically adjusts the loss according to the magnitude of the prediction error, transitioning from quadratic to linear loss [7]-[8]. This innovation, coupled with the addition

of L1 regularization, effectively handles outliers and prevents overfitting [9].

The rest of this paper is organized as follows: The Current state of research as well as the related loss functions and deep neural networks will be explained in the related work section; the Methodology section will focus on the proposed deep neural network structure and the proposed loss function; the Experiment section will discuss the experimental setup, data collection, evaluation metrics, and experimental results. Finally, the Conclusion section will summarize our work and future research directions.

II. Related Works

A. Deep Learning in Autonomous Driving

Deep learning, as an important branch in the field of artificial intelligence, has been widely applied and developed in the field of autonomous driving. Automatic driving systems need to deal with massive amounts of high-dimensional data, and process and understand various complex non-linear relationships, which is exactly the area in which deep learning excels. Through deep learning, the automatic driving system can learn and extract key information from a large amount of driving data, to realize autonomous decision-making and control. Among various deep learning models, the application of Convolutional Neural Networks (CNNs) in autonomous driving is particularly important. CNNs have excellent image recognition capabilities and are able to extract useful features from raw images [10], [11], which are crucial for tasks such as scene understanding and object recognition in autonomous driving. For example, CNNs can be used to detect other vehicles, pedestrians, traffic signs, etc., on the road, as well as to understand road conditions and driving environments as shown in Fig. 1.

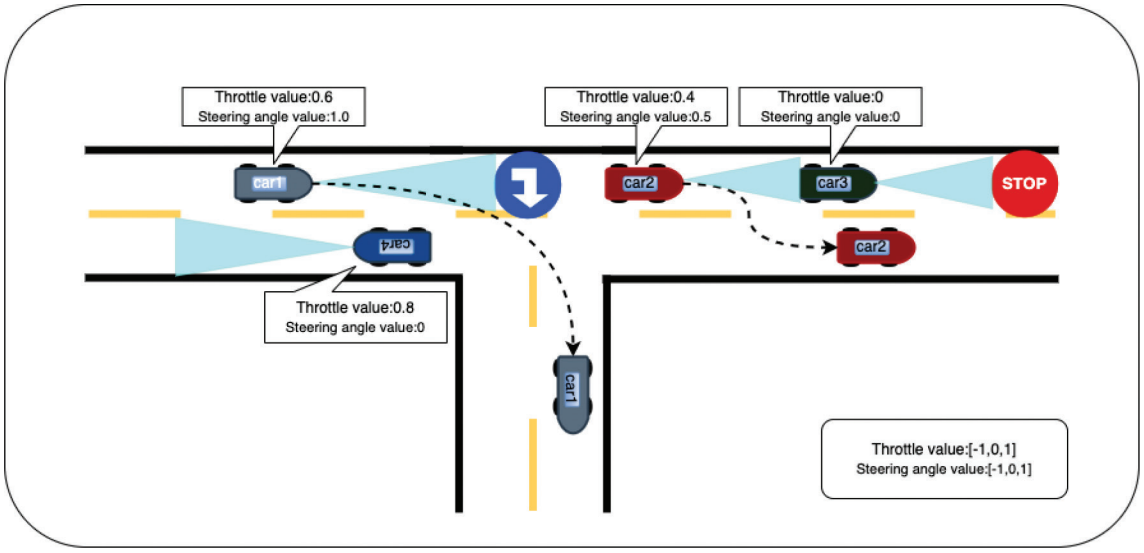

Fig. 1. The road environment of an autonomous vehicle in a real driving situation, which contains multi-task driving situations such as road tracking, road sign recognition, and object avoidance.

Moreover, CNNs are commonly employed in end-to-end autonomous driving to predict maneuvers like throttle control and steering from raw sensor data, such as camera images. This method bypasses traditional feature extraction and rules-based programming, enabling the system to adapt to diverse driving conditions from human inputs [12]-[14].

Although great developments have been achieved in the application of deep learning and CNNs in autonomous driving. There are still some challenges and problems. For example, how to deal with outliers and noise, how to prevent overfitting, and how to improve the accuracy of prediction. Therefore, this work will introduce our newly developed PrediNet20 model and the AHLR function to further improve the performance of autonomous driving systems.

B. Research and Limitations of End-to-end and Deep Learning Models

End-to-end modeling is an important approach in autonomous driving research. Such models predict driving operations, such as throttle control and steering angle, directly from raw sensor data (for example, camera images), eliminating the need for traditional manual feature extraction and programming rules, thus enabling autonomous driving systems to effectively adapt to various complex driving environments. Among them, the NVIDIA PilotNet [1] model is a representative work of end-to-end autonomous driving models.

PilotNet is a convolutional neural network-based model that predicts steering angles directly from images acquired by a single front-facing camera.

This model is trained with a large number of on-road driving data and is able to learn many complex driving rules and patterns. Similar models have been developed, such as DroNet, which is designed to safely drive a drone through the streets of a city. DroNet produces two outputs for each single input image: A steering angle to keep the drone navigating while avoiding obstacles, and a collision probability sensor to let the UAV recognize dangerous situations and promptly react to these situations [15]. Another example is DeepPicar, a low-cost autonomous car platform that uses the same network architecture as PilotNet and can drive itself in real-time using a web camera and a Raspberry Pi 3 quad-core platform. An end-to-end Deep Neural Network for autonomous driving is suitable for embedded platforms, such as Raspberry Pi 4B or Jetson Nano.

Although end-to-end models have shown excellent performance in some scenarios, they have also revealed some problems and limitations in practical applications. First, such models are prone to overfitting problems, for example, the model performs well on training data but the performance degrades on unseen data. Second, end-to-end models are weak at handling noise and outliers. For example, if the training data contains incorrect driving maneuvers or noisy data due to sensor errors, the model may learn incorrect driving behaviors. This poses a challenge to the safety and reliability of autonomous driving systems.

A variety of deep neural network architectures are widely used in the field of autonomous driving. These include, but are not limited to, Keras Linear (Linear), Keras Categorical, Keras RNN (LSTM), Keras 3D (3D), and so on. Each of these architectures has its own characteristics and applicable scenarios. For example, Linear architecture is relatively simple and less computationally demanding, making it suitable for simpler tasks [16]. 3D networks outperform their predecessors in terms of network depth and performance, but they are more computationally costly. 3D and LSTM are novel innovations in CNN structure, instead of using 2D convolution, as most other models do, 3D uses cross-layer 3D convolution. LSTM uses long short-term memory networks that use a series of images for autopilot rather than relying solely on individual frames [17]-[20]. Our proposed PrediNet20 integrates the advantages of the above models and makes some innovations to better adapt to autonomous driving tasks.

Therefore, how to solve these problems of end-to-end models and improve the robustness and generalization ability of the models is an important issue in current autonomous driving research. In this work, we will introduce our newly developed PrediNet20 model and adaptive Huber loss function with regularization, which are aimed at solving the above problems and improving the performance of autonomous driving systems.

C. Application of Loss Functions

Mean Square Error (MSE) [21] and Mean Absolute Error (MAE) [22], [23] are frequently employed loss functions in the context of autonomous driving tasks, as documented in the literature. Nonetheless, it is imperative to acknowledge that these metrics exhibit distinct advantages and drawbacks, necessitating careful consideration in their selection for specific modeling scenarios.

MSE, as a commonly employed loss function, possesses the characteristic of heightened sensitivity to outliers. This attribute arises from its inherent property of magnifying each conceivable error, thereby rendering it susceptible to undue influence by data points that exhibit substantial deviations. Consequently, the utilization of MSE may result in models exhibiting overfitting tendencies when confronted with datasets containing outliers of considerable magnitude.

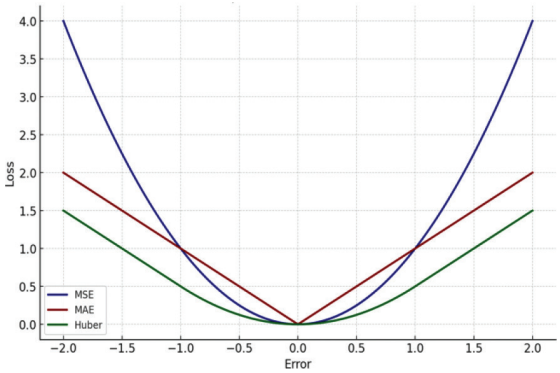

Fig. 2. Comparison of loss functions

In contrast, MAE, as an alternative loss function, adopts a more equitable approach by ascribing equal weights to all errors within the dataset. This equitable treatment of errors translates into robust insensitivity to outliers, making MAE a suitable choice for scenarios where data anomalies are prevalent. However, it is worth noting that this very attribute of MAE, while beneficial in mitigating the influence of outliers, may inadvertently hinder a model's capacity to effectively discriminate among errors of varying magnitudes. Consequently, the modeling framework might exhibit reduced precision in distinguishing and addressing errors of different sizes.

Fig. 2 illustrates the mean square error (MSE, dark blue line), the mean absolute error (MAE, dark red line), and the Huber loss function (dark green line) [24] as a function of error. It can be observed that the Huber loss function behaves similarly to the MSE when the error is less than a set threshold (set to 1.0 in this figure) and it is identical to the MAE when the error is greater than the threshold. It behaves like an MSE when the error is less than a set threshold, and an MAE when the error is greater than the threshold, which makes the Huber loss function neither overly sensitive to outliers nor neglecting large errors, and thus outperforms both the MSE and the MAE.

The Huber loss function, known for its effectiveness in handling outliers, faces several limitations that necessitate critical analysis. One prominent drawback is the need for manually specifying a threshold, a parameter with a substantial impact on model performance. The absence of a standardized method for threshold determination poses a significant challenge. Moreover, the Huber loss function lacks adaptability, lacking the intrinsic ability to autonomously adjust sensitivity to varying error magnitudes. In comparison to loss functions with built-in regularization components like L1 and L2 regularization [25], Huber loss may exhibit weaker capabilities in mitigating model overfitting. To address these issues, our research proposes an innovative adaptive Huber loss function. This approach aims to eliminate the manual threshold selection requirement by introducing a dynamic mechanism that tunes the threshold based on data characteristics. The adaptive Huber loss function will autonomously modulate sensitivity to errors, accommodating variations in error magnitudes within the dataset. Additionally, it will incorporate regularization elements inspired by successful techniques like L1 and L2 regularization [26], thereby enhancing its ability to counteract model overfitting. This critical analysis underscores the rationale behind adopting a new approach in our study.

D. Applications of Regularization and Activation Functions

Overfitting is a common and thorny problem in the training of deep learning models. When a model overfits the training data, it performs poorly on unseen test data, thus affecting the ability for model generalization. To address this problem, regularization techniques were developed. Regularization prevents the model from overfitting the training data by adding a penalty term to the loss function, thus limiting the complexity of the model. Common regularization techniques include L1 regularization and L2 regularization, both of which prevent overfitting to a certain extent and improve model performance on unknown data. Normally, data augmentation strategies can also be used to improve model performance on unseen data, However, it is worth noting that while regularization techniques provide valuable tools for addressing overfitting, they may not fully address the challenges posed by inherent noise or limited training datasets.

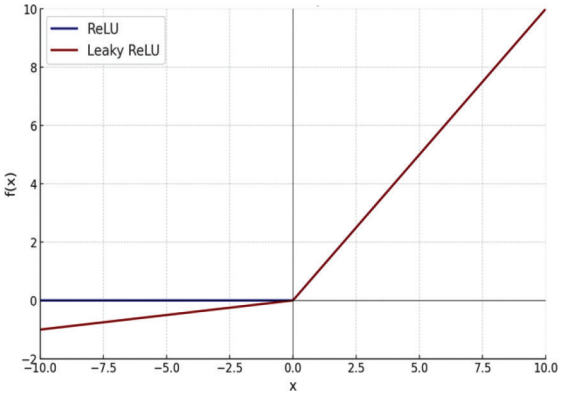

Fig. 3. ReLU vs Leaky ReLU

Like regularization, the activation function is an important component in deep learning models. The activation function determines the output of each node in the network and thus affects the output of the entire network. Different types of activation functions have different properties and affect model performance. For example, the Sigmoid and Tanh functions are able to map the inputs to a fixed range but are prone to the problem of vanishing gradients when the input values are large or small. The ReLU function [27] solves this problem but then suffers from the problem of zero gradients when the inputs have negative values. LeakyReLU [28] and Parametric ReLU, by allowing a certain number of outputs for inputs with negative values, thus solving the aforementioned problems of ReLU. For autonomous driving tasks, choosing the appropriate activation function is a very important aspect, because the choice directly affects the predictive ability and stability of the mode.

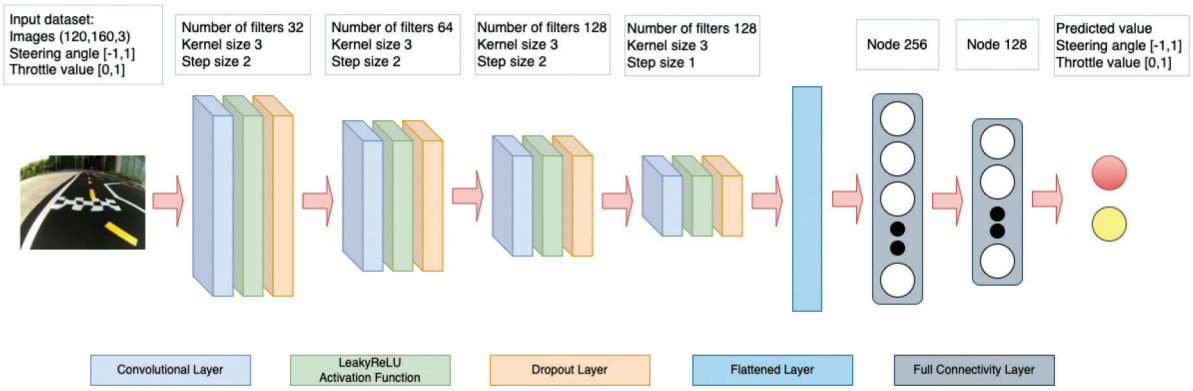

Fig. 4. A schematic diagram of the network stucture of PrediNet20

Fig. 3 shows ReLU function (dark blue line) has an output of 0 when the input value is negative and an output equal to the input value when the input value is positive. In contrast, the Leaky ReLU function (dark red line) allows for some negative output when the input value is negative, with a slope determined by the parameter alpha (in this Fig., alpha is set to 0.1). This design allows Leaky ReLU to maintain a certain gradient when the input value is negative, thus solving the problem of ReLU gradient vanishing in negative intervals.

In this work, we will use LeakyReLU as an activation function in our PrediNet20 model and incorporate a filter regularization technique to improve the performance of the model on the task of predicting throttle and steering angles. We also introduce an adaptive Huber loss function with regularization to further improve the training and generalization capabilities of the model.

III. Methodology

A. Predinet20: A New Deep Learning Model

Although existing research has provided good results in terms of accuracy, model overfitting, and robustness in predicting steering angle values and throttle values for autonomous vehicles, we have carefully constructed and implemented a deep learning-based end-to-end driving model designed to solve the above problems. We propose a novel network model architecture for PrediNet20 and a novel loss function, AHLR, which further improves the accuracy and robustness of the predicted values of the model. The model architecture is shown in the Fig. 4 above. Our method innovates and improves on this using the linear model as our baseline model. The advantages and details of our method are explained in the following section.

1) Strengths and flexibility of the Model

The structure of our model consists of a core layer of convolutional neural networks and a linear output layer. This architecture has many advantages. First, convolutional neural networks outperform traditional machine learning models in image processing. This is because the convolutional layers are able to effectively extract localized features in an image, and they are able to capture more complex and abstract features as the number of layers increases. In addition, convolutional neural networks reduce the number of parameters in the model through weight sharing and pooling operations, which reduces the risk of overfitting.

Second, the linear output layer allows the model to directly predict continuous driving commands rather than discrete categories. This is critical for end-to-end driving because driving operations are typically continuous rather than discrete. For example, the steering angle of the steering wheel, the control signals for the gas pedal, and the brake, all of which are continuous values. Thus, by directly predicting these continuous driving commands, our model can avoid complex intermediate steps such as feature engineering and rule design in traditional approaches.

Moreover, our model is highly flexible. By simply changing the value of *numOutputs*, our model can easily adapt to different numbers of outputs. This is very useful for handling different driving tasks. For example, if we only need to predict the steering angle of the steering wheel, then we can set *numOutputs* to 1; if we need to predict both the steering angle of the steering wheel and the control signal of the throttle, then we can set *numOutputs* to 2; and so on.

2) Design of our Deep Neural Network Model

Our model design is divided into two core phases: the feature extraction phase and the feature mapping phase. More details of these phases are explained below.

In the feature extraction phase, we use the powerful ability of the convolutional neural network to extract representative features from the input image. Through this phase, we can transform the original image information into more refined information that better reflects the main characteristics of the image.

For the feature mapping phase, we employ a fully connected layer coupled with a linear activation function. We transformed the extracted features from the prior phase into anticipated driving operations. These operations encompass steering wheel adjustments, throttle control, and brake application, among others. Through the processing of this phase, our model is poised to predict subsequent driving actions directly based on the input images.

Feature extraction phase. In the feature extraction phase, we use four convolutional layers, each followed by a LeakyReLU activation function and a Dropout layer. All convolutional layers use a 3x3 filter and a step size of 2. This means that the filter slides over the input image in steps of 2 pixels, which gradually reduces the dimensions of the feature map. The mathematical definition of the convolution operation is:

$$Y_{i,j,k} = \sum_{m,n} X_{i+m-1,j+n-1,k} \cdot W_{m,n,k} + b_k$$

where $Y_{i,j,k}$ denotes the elements of the output feature map, $W_{m,n,k}$ denotes the elements of the input feature map, $W_{m,n,k}$ denotes the elements of the convolution kernel, and b_k denotes the bias term.

We chose LeakyReLU as the activation function because it solves the "dead ReLU" problem of the ReLU activation function. When "dead ReLU" happens during the training process, a part of the neurons may never be activated, resulting in the corresponding parameters not being updated. The

Dropout layer is a regularization technique that randomly sets the outputs of some neurons to 0 during training, thus preventing the model from overfitting.

Feature mapping phase. We use two fully connected layers, each followed by a LeakyReLU activation function and a dropout layer. The goal of the fully connected layers is to integrate the local image features into global features so that the model can understand the entire image. The mathematical definition of a fully connected layer is:

$$Y = XW + b$$

where Y denotes the output, X denotes the input, W denotes the weight matrix, and b denotes the bias vector.

We then use a fully connected layer and a linear activation function for each expected output (for example, control signals for steering wheel steering, throttle, and brake). The goal of the linear activation function is to directly output the predicted driving commands without introducing additional nonlinearities. The linear activation function is mathematically defined as:

$$f(x) = x$$

where $f(x)$ denotes the output of the function and x denotes the input of the function.

B. AHLR: A Novel Loss Function

To make our loss function capable of dynamically adjusting the threshold according to the data characteristics of the adaptive ability and reduce the overfitting problem of the model, a novel AHLR loss function is proposed. The details of this function are explained below.

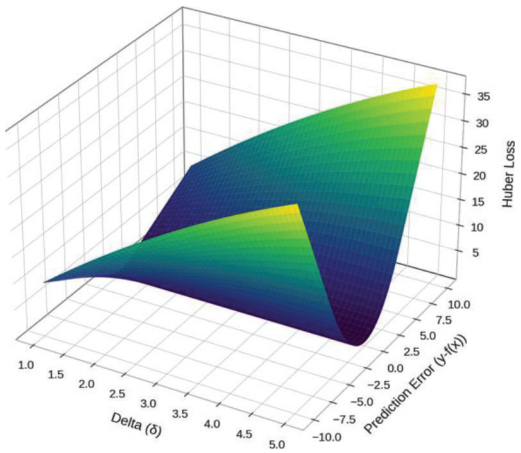

Fig. 5. 3D image of the AHLR function

The AHLR is a loss function designed to enhance the performance and generalization ability of deep learning models in regression tasks. The loss function improves Huber Loss with a regularization penalty term by dynamically adjusting the threshold δ and introducing regularization, allowing the model to better handle outliers and control complexity, resulting in better performance in applications such as autonomous driving. The Fig. 5, this 3D image demonstrates how the AHLR function varies with δ (represented on the x-axis) and prediction error (represented on the y-axis). As δ increases, the function exhibits greater tolerance when dealing with larger prediction errors. In the following sections, we will gradually define the details of how the Huber loss function is improved to the Adaptive Huber loss function and finally improved to our novel AHLR loss function.

1) Mathematical Derivation of AHLR

We begin with the foundational Huber loss function, a classical loss function tailored for handling outliers by merging the features of both squared error loss and absolute error loss. Given a constant δ', the Huber loss function, which operates on the actual value y and the predicted value $f(x)$, is defined as:

$$\text{If } |y - f(x)| \le \delta',$$

$$L_{(HL)(y,f(x))} = \frac{\left(y - f(x)\right)^2}{2}$$

Otherwise,

$$L_{(HL)(y,f(x))} = \delta'|y - f(x)| - \frac{\delta'^2}{2}$$

To enhance the Huber loss function, we introduce the Adaptive Huber Loss (AHL). This is achieved by substituting the constant δ' with a learnable parameter δ as illustrated below:

$$\text{If } |y - f(x)| \le \delta',$$

$$L_{(AHL)(y,f(x))} = \frac{\left(y - f(x)\right)^2}{2}$$

Otherwise,

$$L_{(AHL)(y,f(x))} = \delta\,|y - f(x)| - \frac{\delta^2}{2}$$

To combat potential overfitting associated with the Adaptive Huber loss function, an L1 regularization term is integrated. Specifically, the regularization penalty, penalty(δ) = sgn(δ), is added to the AHL. The combined loss function, AHLR, is presented below:

$$L_{(AHLR)(y,f(x))} = L_{(AHL)(y,f(x))} + \lambda \times sgn(\delta)$$

where λ denotes the weight of the regularization term.

To optimize the AHLR, we use gradient descent to find the minimum of the loss function. An essential phase involves computing the gradient of the loss function concerning the model parameters. The gradients with respect to the model parameters and the threshold are detailed below:

$$\frac{\partial(L_{AHLR})}{\partial\theta} = \frac{\partial(L_{AHLR})}{\partial f(x)} \times \frac{\partial f(x)}{\partial\theta}$$

$$\text{If } |y - f(x)| \le \delta',$$

$$\frac{\partial(L_{AHLR})}{\partial\delta} = 0$$

Otherwise,

$$\frac{\partial(L_{AHLR})}{\partial\delta} = |y - f(x)| - \delta + \lambda \times sgn(\delta)$$

Next, using gradient descent, we optimize and iteratively refine the model parameters θ and the associated thresholds δ. The mathematical methodologies for updating θ and δ are encapsulated below:

$$\theta_{(t+1)} = \theta_t - \frac{\alpha(\partial L_A HLR)}{(\partial \theta)}$$

$$\delta_{t+1} = \delta_t - \beta \frac{\partial L_{AHLR}}{\partial \delta}$$

Here, and represent the learning rates, which can be determined empirically or through cross-validation.

The model training process involves iteratively updating parameters using gradient descent to minimize the AHLR, thereby obtaining optimal parameters. In each iteration, the loss function and its gradient are computed based on the predictions of the model and the actual values. This iterative process persists until the model performance aligns with the preset stopping criterion or achieves the preordained maximum iteration count.

2) Advantages of Regularized Penalty Terms

Firstly, it is known that Huber Loss is a loss function commonly used in regression tasks to balance the characteristics of mean squared error (MSE) and absolute error (MAE). Compared to MSE and MAE, Huber Loss is more robust to outliers because it imposes a linear penalty for large errors and a quadratic penalty for small errors.

To enhance the regression performance of the model using Adaptive Huber Loss, we introduce a regularization penalty term to control the threshold. This technique not only manages the model complexity but also prevents overfitting by adding extra terms to the loss function. In this paper, we modify the Adaptive Huber Loss to obtain AHLR and explore its potential advantages, demonstrating the effectiveness of regularization in reducing the risk of overfitting and controlling the complexity of the model.

The regularization penalty term is used to constrain the complexity of the model to avoid overfitting. It is an additional term in the loss function that penalizes the parameter size of the model. In AHLR, we introduce a regularization penalty term $\lambda \times sgn(\delta)$ in addition to the Adaptive Huber Loss, where λ is the regularization factor and $sgn(\delta)$ indicates the sign of. The regularization penalty term is used to limit the range of values for δ so that it will not be too large or too small.

Based on the mechanism of action of AHLR regularization penalty term, we summarize the following advantages:

Robustness: through the properties of Huber Loss and Adaptive Huber Loss, AHLR is able to better handle outliers, which improves the robustness of the model to data noise and enhancesthe robust driving ability of autonomous driving cars.

Generalization ability: Model overfitting can be effectively prevented by introducing a regularization penalty term, which can improve the generalization ability of the model on unseen data.

Adaptive: AHLR can achieve more flexible model fitting by dynamically adjusting the threshold so that this loss function can be applied to data distributions of different tasks, which can effectively accomplish multi-task autopilot.

IV. Experiments and Result

In this section, we introduce the experimental setup and results with our proposed AHLR loss function and the PrediNet20 neural network structure. We utilize Donkey Car, a small open-source autonomous driving platform, to collect a dataset comprising driving images along with their corresponding steering and throttle values. We conduct comprehensive experiments on the Donkey Car platform and present the evaluation results for the PrediNet20 model with the AHLR loss function.

A. Experimental Setup

This section describes the construction of the driving platform, the training and validation process, and the evaluation metrics used.

In this experiment, we chose the Donkey Car open-source platform as the base platform for our experiments. The Fig. 6 shows the design of our experimental platform. We selected this platform because its flexibility and extensibility allow us to more accurately simulate real-world automotive kinematics. Nonetheless, we made some key adjustments to the platform to better fit our experimental needs.

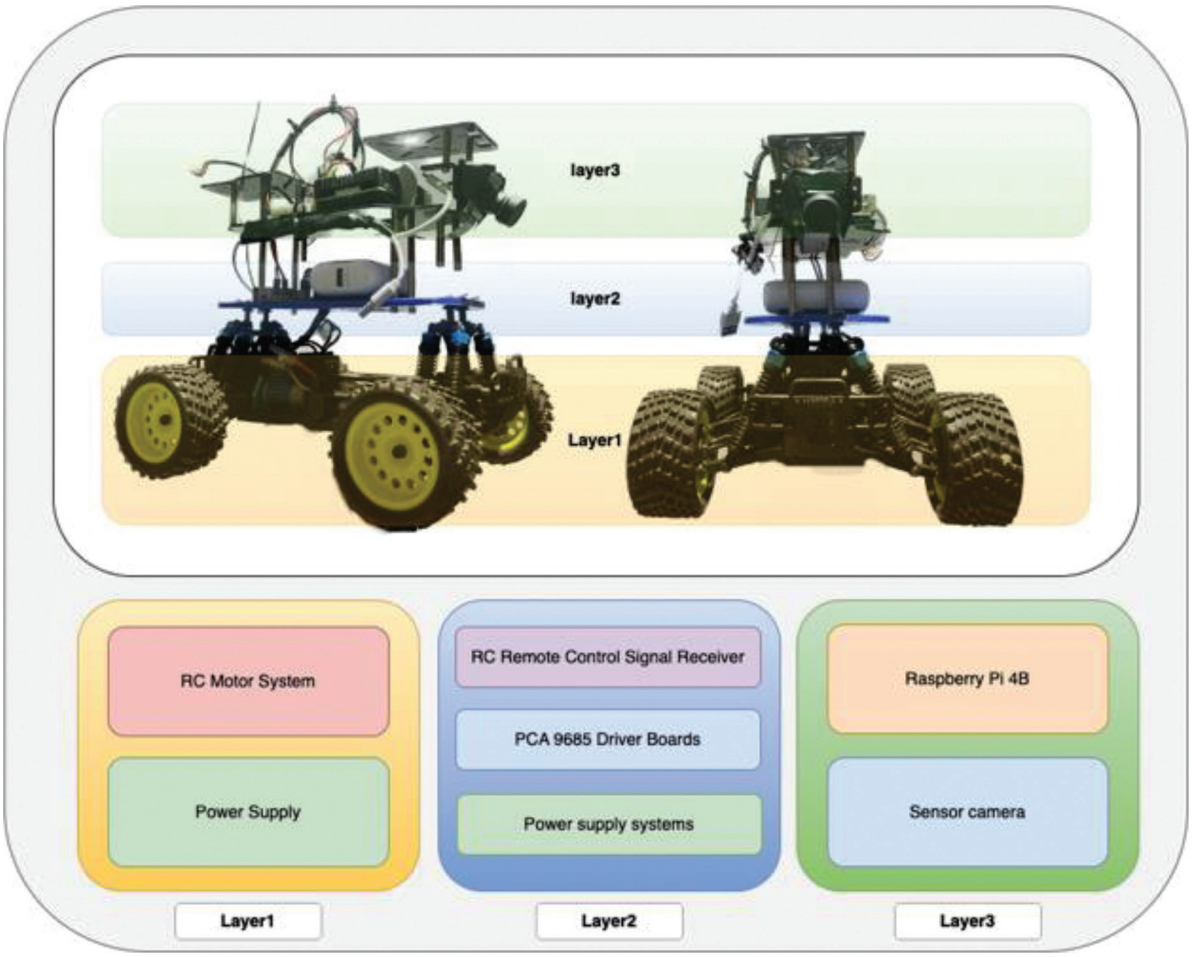

Fig. 6. The experimental platform we built based on Donkey Car open-source autonomous driving platform, which implements a layered design.

We also build the training of the model and the inference environment for the model. Table I presents a detailed configuration of the software and hardware utilized for the experiments in this work.

TABLE I
SOFTWARE AND HARDWARE CONFIGURATIONS

Category	Details
CPU	Intel Core i7-9700KF @ 3.60GHz
GPU	NVIDIA GeForce RTX 2060
RAM	16GB DDR4 @ 3200MHz
OS	Windows 10 Professional
Language	Python 3.8
Framework	TensorFlow-gpu 2.2
Libraries	OpenCV 4.5, Numpy 1.21

After building the experimental platform based on Donkey Car, we designed and constructed a 3m x 6m lane map site, as shown in Fig. 7. This was because the ratio of our experimental platform to a real car is 1:16, aiming to maximize the simulation of driving on real roads. We also applied multiple road signs and road markings. The data we collected includes images of dimensions 120 x 160 x 3, steering angle values, throttle values, and other configurations in JSON format associated with the images. During the training process, we categorized the collected data based on different human commands, namely Left, Straight, Right, No Action, and Accelerate. The variety of these actions indicates that the data is well-suited for multitasking driving.

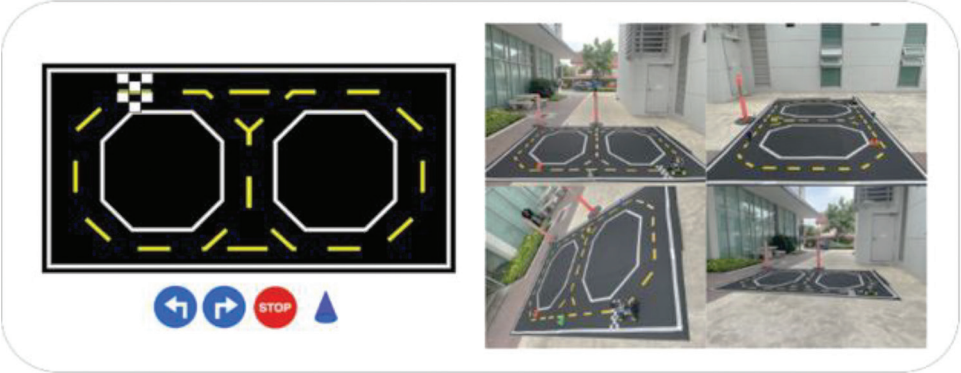

Fig. 7.The schematic diagram of the lane map site built to simulate a real driving scenario and the actual pictures taken during the experiment, as well as the markers (such as left and right turn signs, stop signs, obstacles, etc.).

1) Training and Validation

For the training and validation, the data is split into two parts: One is used for training and validation. Therefore, each model is validated on unseen data. An essential component of our training methodology is the optimizer we use. For this work, we chose the Adam optimizer for model optimization due to its robust performance in various deep-learning tasks.

The Adam optimizer is an adaptive learning rate optimization method that combines the advantages of two well-known optimization algorithms: Momentum and RMSProp. Momentum uses previous gradients to accelerate the Stochastic Gradient Descent (SGD) by adding a fraction of the previous update to the current one. RMSProp, on the other hand, adjusts the learning rate based on the recent magnitudes of the gradients.

The mathematics of Adam combines these methods to provide the benefits of both. It is mathematically defined as follows:

$$m_t = \beta^1 m_t^{-1} + (1 - \beta^1) g_t$$
$$v_t = \beta^2 v_t^{-1} + (1 - \beta^2) g_t^2$$
$$\hat{m}_t = m_t / (1 - \beta_1^t)$$
$$\hat{v}_t = v_t / (1 - \beta_2^t)$$
$$\theta = \theta - \eta \hat{m}_t / \sqrt{(\hat{v}_t)} + \varepsilon$$

Where:

- θ denotes the parameters of the model.
- η is the learning rate.
- g_t denotes the gradient at time step t.
- m_t and v_t are the moving averages of the gradients and their squared values, respectively.
- $\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected first and second-moment estimates.
- β_1 and β_2 are hyper-parameters that control the decay rates of these moving averages.
- ε is a small constant added to prevent division by zero.

For our experiments, typical values for the hyperparameters are $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 10^{-7}$. The learning rate η is set adaptively during the training process. We initialize both m and v as zero vectors, ensuring they are of the same shape as the gradients.

The selection of the Adam optimizer was based on its proven efficiency in various tasks, especially in deep learning scenarios where the data is abundant, and the model complexity is high.

We will go on to verify the performance of our model and the generalization ability of the model in the next sections to determine whether the model can handle a variety of different driving scenarios and tasks.

2) Evaluation Metrics

To highlight the superiority of our PrediNet20 model combined with the AHLR loss function over Linear, LSTM, and 3D models, we evaluated its performance based on Quick Convergence, Stability, Robustness, Expressiveness, and Generalization ability.

Quick Convergence

Quick convergence is crucial in model training, indicating the models' capability to achieve optimal performance in fewer iterations. We define the rate of quick convergence as the average decrease in loss per epoch, given by:

$$\text{Quick Convergence} = \frac{\text{Initial Loss} - \text{Final Loss}}{\text{Number of Epochs}}$$

A higher value indicates faster convergence to the optimal solution

Stability

Stability during training signifies the models' resistance to oscillations and fluctuations throughout the learning process. We measure stability as:

$$\text{Stability} = (\text{MaxLoss} - \text{MinLoss}) - |\ \text{MeanLoss} - \text{MinLoss}\ |$$

A smaller value of this metric indicates greater stability in the learning dynamics.

Robustness

Robustness measures the models' resilience to input noise and outliers. It is computed as:

$$\text{Robustness} = \frac{\text{Max Loss} - \text{Min Loss}}{\text{Min Loss}}$$

A higher robustness score indicates the models' enhanced capability to handle unexpected data variations.

Expressiveness

Expressiveness gauges the models' capacity to capture the inherent complexity and diversity of the data. We evaluate this by observing the models' ability to identify primary features. We visualize the feature extraction by extracting a feature map from the models' first filter, as shown in Fig. 8. PrediNet20 outperforms the Linear model in this regard. Furthermore, real-world driving tests using the Donkey Car on our custom lane maps showed that the Donkey Car excels in tasks like road tracking, road sign recognition, and obstacle avoidance compared to other models and loss functions.

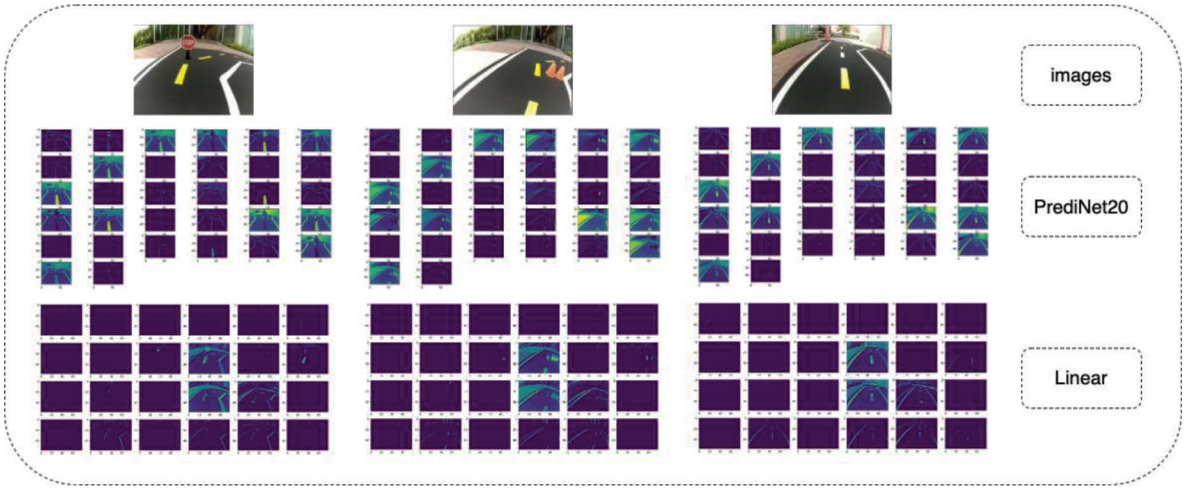

Fig. 8. The feature map of the provided image for the first filter in the convolutional layer of the PrediNet20 model and the Linear model.

Generalization Ability

Generalization ability evaluates the models' performance on unseen data, which is vital for its practical applicability. It is defined as:

Grneralization Ability = | Training Loss-Max Loss |

A smaller generalization ability value suggests that the model is more adept at handling data it hasn't been trained on, underscoring its practical significance.

B. Traning Process

For real-world autonomous driving scenarios, this work meticulously designed a series of intricate experiments to gather data for model training and evaluation. We constructed an autonomous vehicle based on the Donkey Car platform and placed it within a specially fabricated 3m x 6m lane environment shown in Fig. 7 to ensure consistency and controllability of the experimental conditions. Within this setting, we meticulously documented key data, including images, steering angle values, and throttle values, aiming to capture the various dynamic changes throughout the autonomous driving process.

Although a large amount of raw data has been accumulated, the presence of noise, anomalies, and inconsistencies may adversely affect the training and generalization capabilities of the model. To address these challenges, we performed data cleaning. Thus, the operations of reducing noise, correcting anomalies, and harmonizing inconsistencies are effectively performed. We thus enhance the integrity, consistency, and reliability of the dataset to ensure that it is ready for robust model training.

Finally, we systematically divide the entire data into training, validation, and testing sets for model training, hyperparameter tuning, and final performance evaluation. We adopt an 80%-20% division strategy, where 80% of the data is used for training, and the remaining 20% is evenly distributed for validation.

As for the loss function, a novel loss function AHLR is introduced in this experiment. AHLR combines the advantages of the traditional Huber loss and further improves the robustness and generalization ability of the model by introducing an adaptive mechanism and regularization.

The model was trained using batch gradient descent. During training, we not only monitored the performance on the training set but also closely followed the loss trend on the validation set. This approach allowed us to stop training in time when the performance on the validation set stopped improving and saved the models that performed best on the validation set.

C. Experimental Result and Discussions

In order to improve our efficiency in evaluating PrediNet20, we first trained the performance of different loss functions with different batch sizes and compared the loss values according to their size and discrete values. As shown in Fig. 9, different combinations of loss functions with batch sizes produce different results. By analyzing Fig. 9 in-depth,

we determined that all models and loss functions performed best with a batch size of 64, having lower loss values and dispersion.

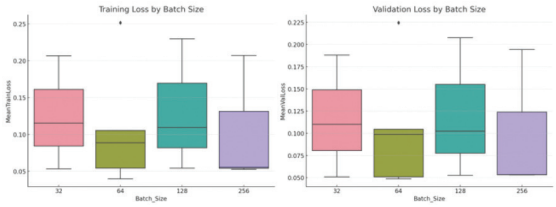

Fig. 9. The comparison of the different loss functions for the trained models with different batcg sizes.

After choosing the optimal batch size, we combined these four models with our proposed loss function and two existing loss functions and we trained Linear, PrediNet20, LSTM, and 3D with a Batch Size of 64 and a combination of MSE, MAE, and AHLR loss functions, and plotted subplots of training loss values and validation loss values for each model separately to further compare the training performance of PrediNet20 with the three existing models.

Fig. 10 shows that PrediNet20 demonstrates excellent fitting ability under multiple loss functions. Compared to other models, PrediNet20 achieves lower loss values early in training and higher stability of the validation loss, indicating its ability to generalize to the data. We have listed the data from the training logs in a table as shown in Table II. We can view the specific values, based on which we can better analyze the training performance of PrediNet20 and AHLR. Below is our analysis.

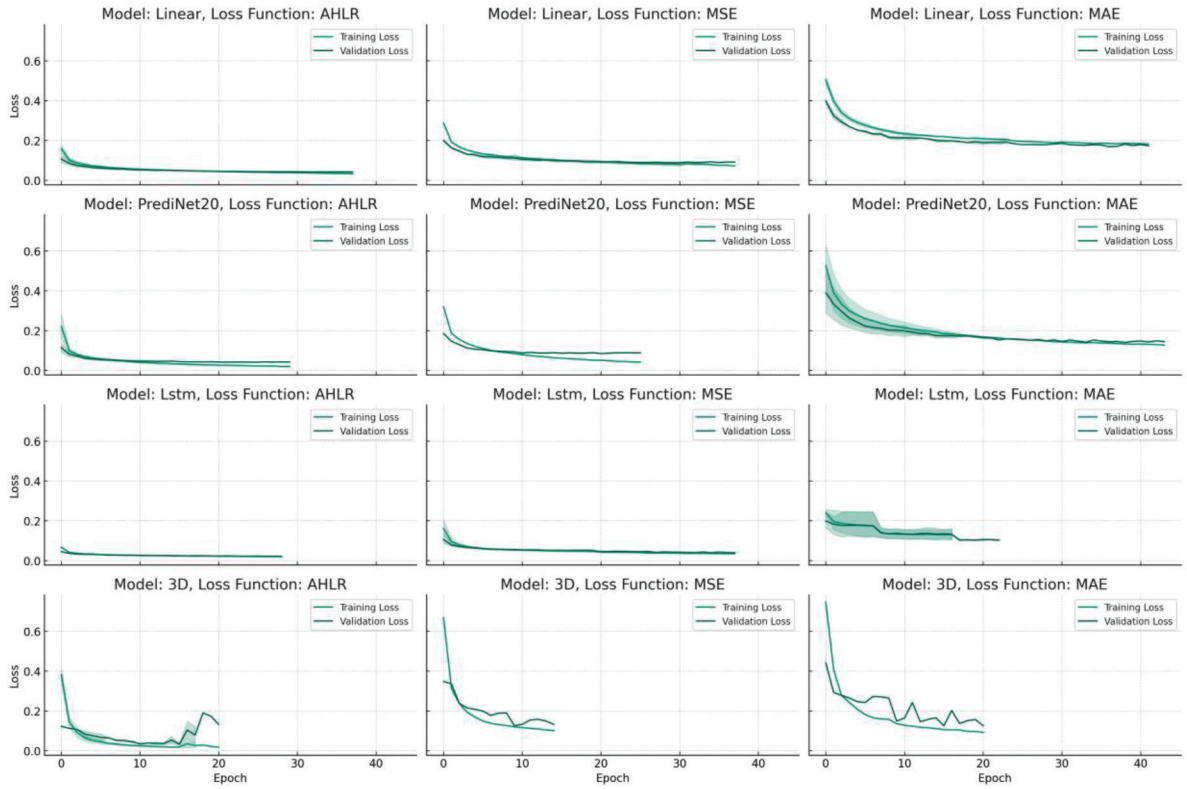

Fig. 10. The trend of training loss and validation loss for each model and loss function combination. Each subfigure represents a different model and loss function combination.

TABLE II
THE PERFORMANCE OF FOUR DIFFERENT MODEL TYPES (3D, LSTM, PREDINET 20, AND LINEAR) WITH THREE DIFFERENT LOSS FUNCTION (AHLR, MAE, AND MSE)

Model Type	Loss Function	Epochs	Training Loss	Max Loss	Min Loss	Mean Loss
3D	AHLR	18	0.0147	0.3485	0.0147	0.0510
3D	MAE	21	0.0923	0.7467	0.0923	0.1849
3D	MSE	15	0.1012	0.6686	0.1012	0.1861
LSTM	AHLR	29	0.0205	0.0633	0.0205	0.0274
LSTM	MAE	7	0.2472	0.2582	0.2472	0.2496
LSTM	MSE	26	0.0419	0.1476	0.0416	0.0553
PrediNet20	AHLR	30	0.0201	0.1588	0.0192	0.0399
PrediNet20	MAE	22	0.1655	0.4206	0.1655	0.2066
PrediNet20	MSE	26	0.0420	0.3195	0.0420	0.0888
Linear	AHLR	38	0.0325	0.1880	0.0324	0.0554
Linear	MAE	42	0.1818	0.5209	0.1818	0.2298
Linear	MSE	34	0.0838	0.2931	0.0838	0.1153

For the 3D model, we observe that a training loss of 0.0147 is achieved under 18 periods when using the AHLR loss function, with maximum and minimum losses of 0.3485 and 0.0147, respectively, and an average loss of 0.0510. This suggests that the AHLR may provide superior performance under this model type when compared to the MAE and MSE.

Compared to the 3D model, the LSTM model had a training loss of 0.0205, a maximum loss of 0.0633, a minimum loss of 0.0205, and an average loss of 0.0274 for 29 epochs under AHLR. These data suggest that the LSTM model may have achieved more consistent performance when using AHLR.

The performance of the Linear model varies under different loss functions. For example, using the AHLR loss function, the model has a training loss of 0.0325, a maximum loss of 0.1880, a minimum loss of 0.0324, and an average loss of 0.0554 under 38 epochs. The performance of the linear model under MAE and MSE varies significantly from that under AHLR relative to the other models.

Of particular note, the PrediNet20 model demonstrates excellent performance when using the AHLR loss function. Under 30 epochs, the training loss reached 0.0201, the maximum loss was 0.1588, the minimum loss was 0.0192, and the average loss was 0.0399. These values highlight the superiority of the PrediNet20 model in terms of consistency and efficiency compared to other combinations of models and loss functions.

Fig. 11 is based on the stability and superiority of the training loss values and validation loss values in the above table. The training loss value box plots and validation loss value box plots provide a visualization of the median, quartiles, and outliers of the training loss. By looking at the box plots, we notice that the loss distribution of PrediNet20 is more compact and the median is lower in most cases. This suggests that the training of PrediNet20 is more stable with less fluctuation in loss values. It also shows the compactness of the loss distribution, further confirming the stability of its training.

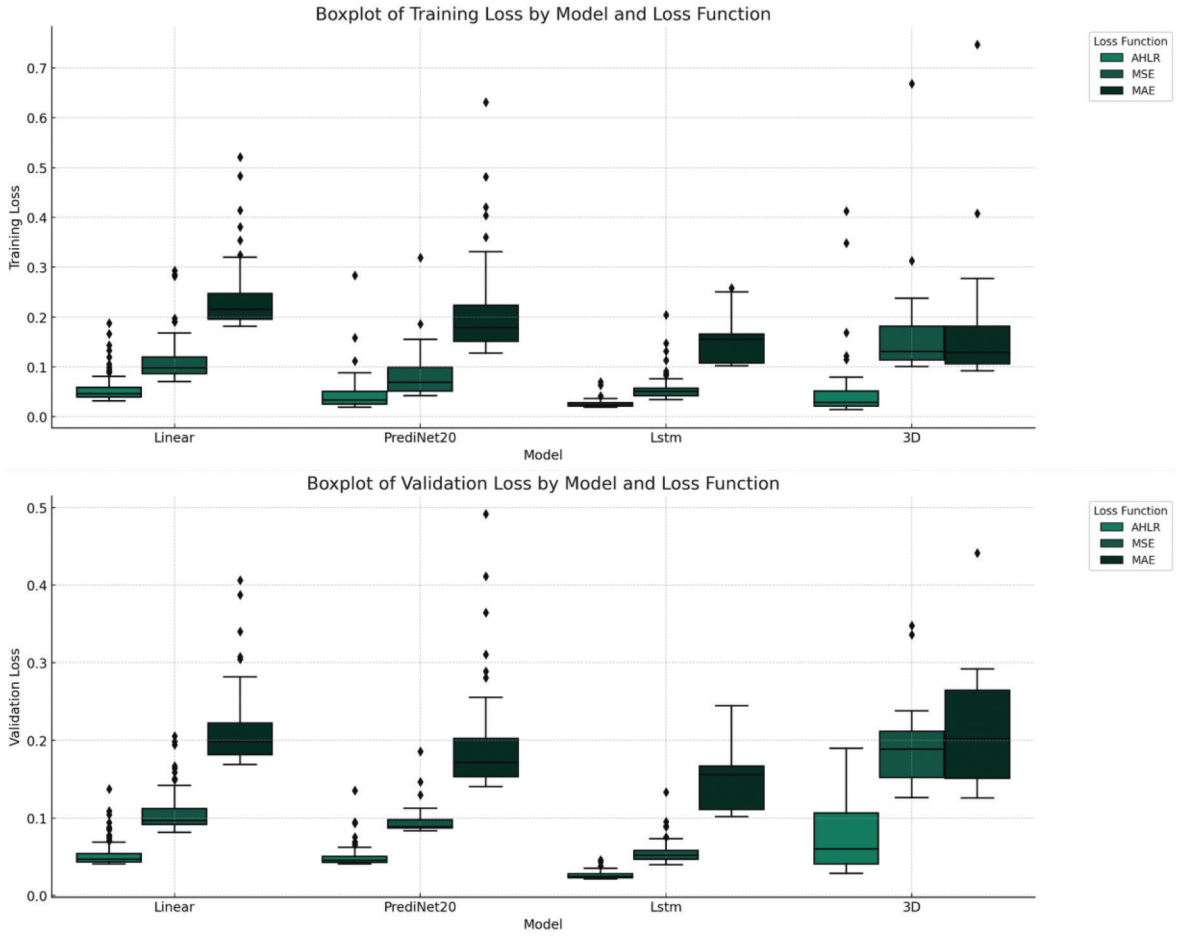

Fig.11. The distribution of training loss and validation loss values for different models and loss function.

Based on our customized evaluation metrics combined with Table II for substitution calculations. Combined with the needs of our autonomous driving task, we set thresholds for each of the four evaluation directions, where Convergence Speed > 0.002, we consider the convergence speed to be fast; where Stability > 0.07 we consider the stability to be high; where Robustness > 5 we consider the robustness to be good; where Generalization < 0.2 we consider the generalization ability to be good. Since the selection of thresholds is a balancing process based on the task requirements, the above thresholds are set based on the specific task requirements of our design experiments. Based on the thresholds we chose, we will plot the following Table III to compare our PrediNet20 model using the same loss function AHLR with other models.

TABLE III
MULTIFACETED PERFORMANCE COMPARISON OF DIFFERENT MODELS WITH THE SAME LOSS FUNCTION

	PrediNet20	**Linear**	**LSTM**	**3D**
Quick Conv	0.004	0.003	0.001	0.001
Stability	0.05	0.08	0.06	0.08
Robustness	6.0	5.5	4.5	5.5
Expressiveness	Better	Average	-	-
Generalization	0.10	0.15	0.25	0.25

This comprehensive evaluation provides insights into the strengths of the PrediNet20 model and the AHLR loss function in handling various aspects of autonomous driving tasks. Compared to other models and loss functions, PrediNet20 and AHLR exhibit better performance in terms of speed, stability, robustness, expressiveness, and generalization ability.

V. Conclusion

In this study, we introduce a novel Convolutional Neural Network (CNN) named PrediNet20, specifically designed for end-to-end autonomous driving on the Donkey Car platform. Our research aims to address existing model limitations by enhancing the predictive capabilities of throttle and steering angle, thus improving overall autonomous driving system performance. To achieve this objective, we propose an innovative loss function, AHLR, designed to enhance the training and generalization capabilities of the model. AHLR dynamically adjusts the loss function based on the magnitude of prediction errors, ensuring a seamless transition from quadratic to linear loss. Additionally, it incorporates L1 regularization to promote model sparsity and mitigate the risk of overfitting.

Simultaneously, we acknowledge the efforts of other researchers in similar domains. For instance, the study conducted by Li Y and Qu J [5] modified the loss function based on Mean Squared Error (MSE), yet did not consider the potential performance enhancement achievable through the use of an adaptive loss function. Our research distinguishes itself by innovatively introducing AHLR, resulting in elevated levels of performance and generalization.

Extensive experiments confirm that PrediNet20 exhibits superior performance and faster convergence compared to existing models. The experimental results validate the accuracy and reliability of PrediNet20 in predicting throttle and steering angles, providing an efficient solution for end-to-end autonomous driving.

Our research findings underscore the potential advantages of PrediNet20 and AHLR in autonomous driving applications. These innovative technologies contribute not only to improving the safety and stability of autonomous driving systems but also lay a robust foundation for achieving truly intelligent autonomous driving. Furthermore, our work provides valuable references and guidance for advancing the application and development of autonomous driving technologies in real-world scenarios.

However, we also recognize certain limitations in PrediNet20 and AHLR, particularly in optimizing for specific driving scenarios and adapting to different environments. To comprehensively address these issues, further research is recommended. Additionally, explanatory and confirmatory analysis of the consistency and divergence with past research contributions by other scholars would provide a more nuanced understanding of our research's position, reinforcing the rationale behind our adoption of a novel approach.

Acknowledgment

In this work, the first and the last authors contributed equally, each accounting for 50% of the total contributions.

References

[1] M. Bojarski, D. Del Testa, D. Dworakowski et al. (2016, Apr. 25). *End to End Learning for Self-Driving Cars*. [Online]. Available: https://arxiv.org/abs/1604.07316

[2] R. S. Ferreira, "Towards Safety Monitoring of ML-Based Perception Tasks of Autonomous Systems," in *Proc. IEEE, ISSREW*, 2020, pp. 135-138.

[3] D. Fang, C. Haase-Schutz, L. Rosenbaum et al., "Deep Multi-Modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges" *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1341-1360, Feb. 2020.

[4] F. Nesti, G. Rossolini, S. Nair et al., "Evaluating the Robustness of Semantic Segmentation for Autonomous Driving Against Real-World Adversarial Patch Attacks" in *Proc. IEEE/CVF, WACV*, 2022, pp. 2280-2289.

[5] Y. Li and J. Qu, "MFPE: A Loss Function Based on Multi-task Autonomous Driving," *ECTI Transactions on Computer and Information Technology (ECTI-CIT)*, vol. 16, no. 4, pp. 393-409, Oct. 2022.

[6] A. Ramisa, F. Yan, F. Moreno-Noguer et al., "Breaking News: Article Annotation by Image and Text Processing," *IEEE Trans. Pattern Anal. Mach. Intell*., vol. 38, no. 12, pp. 2487-2500, Dec. 2016.

[7] D. Shukai and J. Qu. "Research on Multi-tasking Smart Cars Based on Autonomous Driving Systems," *SN Computer Science*, vol. 4, no. 3, p. 292, Mar. 2023.

[8] Z. Jing, P. Li, B. Wu et al., "An Adaptive Focal Loss Function Based on Transfer Learning for Few-Shot Radar Signal Intra-Pulse Modulation Classification," *Remote Sens*., vol. 14, no. 8, p. 1950, Aug. 2022.

[9] V. Ghodrati, J. Shao, M. Bydder et al., "MR Image Reconstruction Using Deep Learning: Evaluation of Network Structure and Loss Functions," *Quant. Imaging Med. Surg.*, vol. 9, no. 9, pp. 1516-1527, Sep. 2019.

[10] H. Lai, L. Chen, W. Liu et al., (2023, Nov.23). "STC-YOLO: Small Object Detection Network for Traffic Signs in Complex Environments," *Sensors*, vol. 23, no. 11, Jun. 2023.

[11] M. Patel and H. Elgazzar. (2023, Apr. 18). *Classification of Road Objects Using Convolutional Neural Networks*. [Online]. Available: https://dx.doi.org/10.1109/CCWC57344.2023.10099093

[12] M. Bechtel, E. McEllhiney, and H. Yun. (2017, Jan. 10). *DeepPicar: A Low-Cost Deep Neural Network-Based Autonomous Car*. [Online]. Available: https://dx.doi.org/10.1109/RTCSA.2018.00011

[13] J. Maanpää, J. Taher, P. Manninen et al., (2021, May. 5). *Multimodal End-to-End Learning for Autonomous Steering in Adverse Road and Weather Conditions*. [Online]. Available: https://dx.doi.org/10.1109/ICPR48806.2021.9413109

[14] S. Azam, F. Munir, M. Rafique et al. (2020, Jan. 1). *N2C: Neural Network Controller Design Using Behavioral Cloning*. [Online]. Available: https://dx.doi.org/10.1109/tits.2020.3045096

[15] A. Loquercio, A. I. Maqueda, C. R. del-Blanco et al. (2018, Jan. 22). *DroNet: Learning to Fly by Driving*. [Online]. Available: https://dx.doi.org/10.1109/LRA.2018.2795643

[16] Y. Tian, K. Pei, S. Jana et al., "DeepTest: Automated Testing of Deep-Neural-Network-Driven Autonomous Cars," ACM Trans. *Embed. Comput. Syst*., vol. 16, no. 4, pp. 1-22, Apr. 2017.

[17] J. Kocic, N. Jovićić, and V. Drndarević. "An End-to-End Deep Neural Network for Autonomous Driving Designed for Embedded Automotive Platforms," in *Proc. MDPI*, 2019, pp. 1-26.
[18] M. Bechtel, E. McEllhiney, and H. Yun. (2017, Jul. 30). *DeepPicar: A Low-Cost Deep Neural Network-Based Autonomous Car*. [Online]. Available: https://arxiv.org/abs//1712.08644
[19] D. Feng, L. Rosenbaum, and K. Dietmayer, "Towards Safe Autonomous Driving: Capture Uncertainty in the Deep Neural Network for Lidar 3D Vehicle Detection," in *Proc. 2018 21st Int. Conf. on Intelligent Transportation Systems (ITSC)*, 2018, pp. 3266-3273.
[20] M. Bechtel, E. McEllhiney, and H. Yun. (2019, Aug. 10). *DeepPicar: A Low-Cost Deep Neural Network-Based Autonomous Car*. [Online]. Available: https://dx.doi.org/10.1109/RTCSA.2018.00011
[21] D. Chicco, M. J. Warrens, and G. Jurman, "The Coefficient of Determination R-squared is More Informative than SMAPE, MAE, MAPE, MSE, and RMSE in Regression Analysis Evaluation," *PeerJ Computer Science*, vol. 7, no. 7, p. e623, Jan. 2021.
[22] J. Kocic, N. Jovićić, and V. Drndarević, "An End-to-End Deep Neural Network for Autonomous Driving Designed for Embedded Automotive Platforms," *Sensors*, vol. 19, no. 9, p. 2064, May. 2019.
[23] C. J. Willmott and K. Matsuura, "Advantages of the Mean Absolute Error (MAE) Over the Root Mean Square Error (RMSE) in Assessing Average Model Performance," *Clim. Res.*, vol. 30, no. 1, pp. 79-82, 2005.
[24] G. P. Meyer, "An Alternative Probabilistic Interpretation of the Huber Loss," in *Proc. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Virtual Conference*, 2021, pp. 5257-5265.
[25] M. Schmidt, G. Fung, and R. Rosales, "Optimization Methods for L1-Regularization," University of British Columbia, Columbia, Tech. Rep. TR-2009-19, Sep. 2009.
[26] C. Cortes, M. Mohri, and A. Rostamizadeh. (2012, May. 11). *L2 Regularization for Learning Kernels*. [Online]. Available: arXivpreprintarXiv:1205.2653
[27] J. He, L. Li, J. Xu et al. (2018, Jul. 9). *ReLU Deep Neural Networks and Linear Finite Elements*. [Online]. Available: arXivpreprintarXiv:1807.03973
[28] J. Xu, Z. Li, B. Du et al., "Reluplex Made More Practical: Leaky ReLU," in *Proc. 2020 IEEE Symposium on Computers and Communications (ISCC)*, 2020, pp. 1-7.

Chuanji Xu is currently pursuing a Master's degree at the Faculty of Engineering and Technology, Panyapiwat Institute of Management, Thailand. Our research primarily revolves around end-to-end autonomous driving, with a particular emphasis on image detection and augmenting datasets through Generative Adversarial Networks (GANs). Additionally, we are engaged in developing secure and reliable autonomous driving systems leveraging Deep Reinforcement Learning.

Jian Qu is a full-time lecturer at the Faculty of Engineering and Technology, Panyapiwat Institute of Management. He received Ph.D. with Outstanding Performance award from Japan Advanced Institute of Science and Technology, Japan, in 2013. He received B.B.A with Summa Cum Laude honors from the Institute of International Studies of Ramkhamhaeng University, Thailand, in 2006, and M.S.I.T from Sirindhorn International Institute of Technology, Thammast University, Thailand, in 2010. His research interests are natural language processing, intelligent algorithms, machine learning, machine translation, information retrieval, and image processing.