Hybrid Transfer Learning and Kalman Filter structured RNN for Autonomous Steering Prediction

1st Shao-Hsuan Huang
dept. Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden
huangs@student.chalmers.se

2nd Yuhong Zhou
dept. Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden
yuhong@student.chalmers.se

***Abstract*—This document explores a method for predicting the steering angle of autonomous vehicles by integrating a transfer-learning-based convolutional neural network (CNN) with a Kalman filter-enhanced recurrent neural network (RNN) structure. The CNN, adapted from VGGNet19, acts as a visual sensor to interpret front-camera images, capturing driving scene dynamics. This input is processed through a Kalman filter framework, complemented by three RNNs, each dedicated to learning and predicting distinct aspects of the system: the covariance matrix of states, the covariance matrix of measurements, and the system model itself. After training, these predictions are combined using the Extended Kalman Filter (EKF) equations to estimate the final steering angle. Our model's performance is comparing with the top-ranked solution from the Udacity self-driving car challenge, from which our dataset is sourced. The comparison is conducted with both real-time and delayed image inputs, revealing that our approach achieves highly accurate steering predictions that we achieve a ranking of number 7 on the leader board, and even outperforms the challenge's best model by maintaining lower loss values under delayed conditions.**

***Index Terms*—kalman filter, VGG19, autonomous vehicles, Convolutional Neural Networks (CNNs), transfer learning**

I. Introduction

System control has been a critical area of research and industry focus, especially for vehicles where the goal is to enable safe, smooth, and eventually autonomous driving.

In traditional vehicle control, we start by identifying the system model. This often requires attaching sensors, selecting specific inputs like step signals, and navigating challenges related to sensor accuracy and model complexity. After estimating the model, a Kalman filter is commonly used to balance predictions and measurements, refining the states. These states then guide controllers, such as linear-quadratic regulators, to drive the system. Although this process can achieve optimal control, it is complex, involves extensive tuning, and depends heavily on accurate measurements.

To handle the complexity discussed, many studies use neural networks as an end-to-end approach [1]. Unlike traditional control methods, which don't improve with more data, neural networks can benefit from larger datasets to boost performance. For instance, a convolutional neural network (CNN) can be trained on a large dataset to predict steering angles from car front camera images. Some approaches also use recurrent neural networks (RNNs) to capture time-based patterns, as driving decisions often depend on both current and previous conditions.

Studies have shown that using neural networks for the entire driving pipeline can be effective [2]. However, these networks are hard to interpret and analyze, making it difficult to choose the best model quickly. Selecting and tuning network parameters takes time and usually requires a trial-and-error approach, adding to the challenge.

Considering the strengths and weaknesses of both methods for steering angle prediction, combining them could bring the best of both worlds. Neural networks could learn how to measure states, set covariance parameters, and identify system models from input data. With more data, their accuracy would improve further.

In this combined approach, neural networks wouldn't act as "black boxes" but would provide interpretable outputs like model and covariance matrices, allowing for easier parameter tuning and adjustments. By integrating these networks with a Kalman filter, we achieve optimal state estimation, merging the learning power of neural networks with the stability of the Kalman filter.

II. Background theory

We started by leveraging a contest organized by Udacity [3], which focused on using deep learning to predict steering angles for self-driving cars. This contest provided a useful dataset and several competitor solutions that aligned well with our goals, serving as a strong foundation for our experiments and development.

The dataset included around 30 minutes of driving images captured from front, left, and right cameras under various driving conditions for training, along with a 280-second unlabeled set of driving images for testing [3].

To optimize our performance, we focused on the top competitor's strategy, which employed transfer learning with a convolutional neural network based on Nvidia's pre-trained model [4]. This inspired us to adopt transfer learning with VGGNet19, which has shown the best performance for steering angle prediction in the ROS2 simulation environment [5].

After predicting the steering angle from images, we also wanted to incorporate time-based relationships using a Kalman filter in the network. While some research has used recurrent

neural networks (RNNs) to predict the Kalman gain from measured outputs and previous states [6], this approach doesn't fully suit our needs, as we aim to predict the system model as well. We instead adopted an approach from research that utilizes RNNs to predict both covariance matrices and the system model [7], which aligns well with our objectives.

III. Proposed Solution

A. Network Construction

We applied transfer learning from VGGNet19 to create a convolutional neural network "sensor" for steering angle prediction from images. Using PyTorch, we loaded the pre-trained VGGNet19 (trained on ImageNet) and replaced its classifier with the recommended structure from [5], summarized below.

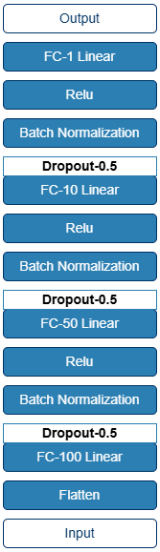


Fig. 1. VGGnet19 classifier

The convolutional neural network processes the current image input and outputs a steering angle prediction for that exact moment.

To predict the system model and covariance matrices, we adopted the method from [7] using Long Short-Term Memory (LSTM) networks. LSTMs, a type of recurrent neural network, effectively handle long-term dependencies and mitigate the vanishing gradient problem through gated mechanisms. Our LSTM models include additional dropout layers and fully connected layers, summarized as follows:

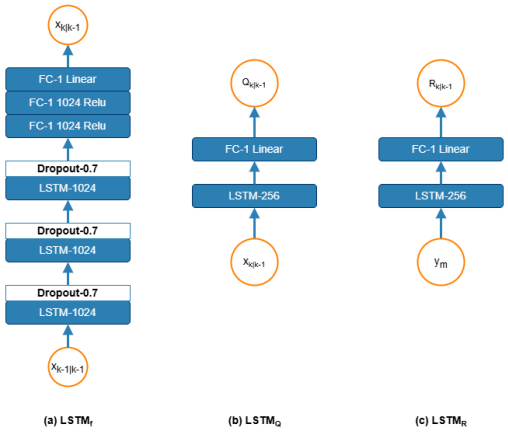


Fig. 2. Long Short Term Memory networks

To incorporate time-wise relationships, each LSTM receives a sequence of 10 previous time steps rather than a single time-step prediction or update. This enables the LSTMs to capture the temporal dependencies between states effectively. If there are fewer than 10 previous steps available (e.g., at the start), the remaining inputs are padded with zeros.

The complete setup of all LSTMs will be detailed in the Equations section.

B. Equations

Given the potentially nonlinear nature of the vehicle system, we use an Extended Kalman Filter (EKF) to estimate the states. The estimation process begins with the prediction step, as outlined in the following equation:

$$x_{k|k-1} = f(x_{k-1|k-1}) \tag{1}$$

$$y_{k|k-1} = x_{k|k-1} \tag{2}$$

$$P_{k|k-1} = FP_{k-1|k-1}F^T + Q_{k|k-1} \tag{3}$$

In this setup:

- Subscripts $k|k-1$ use information from the previous step to predict the current movement, while $k-1|k-1$ updates the previous movement with prior information.
- y: The system output, which is the predicted steering angle.
- x: Represents the states, including the steering angle, aligning state and output, so the usual H matrix (mapping states to outputs) is simply an identity matrix.
- The model describes the relationship between the previous time update and the current prediction.
- f: The system model predicted by the LSTM; F: its linearized form.
- P: Covariance matrix of state error; Q: Covariance matrix of the states, also predicted by an LSTM for Q.

This approach allows the LSTM to dynamically capture both the system model and the associated covariances.

Here is the definition for the update part:

$$K_{k|k} = P_{k|k-1}(P_{k|k-1} + R_{k|k-1})^{-1} \tag{4}$$

$$x_{k|k} = x_{k|k-1} + K_{k|k}(y_m - y_{k|k-1}) \quad (5)$$

$$P_{k|k} = (I - K_{k|k})P_{k|k-1} \quad (6)$$

Here is the description with bullet points:

- K: Kalman filter gain, balancing measurement and state prediction.
- R: Measurement covariance matrix, predicted by the LSTM.
- y_m: Measurement from the VGGNet19 network.
- $x_{k|k}$: Final update, predicting the steering angle at time k.

C. Data Preprocessing

After initial testing, we found that extensive data augmentation did not significantly improve model performance, so we opted for simple augmentations like normalization, random flipping, and color jittering. Since we are using VGGNet as the convolutional neural network, resizing images to 224x224 is still necessary.

In training, we use five datasets provided with the contest's website and follow the data extraction procedure outlined in the competition guidelines. We customize a dataset class to efficiently link images with corresponding metadata, and use dataloaders to simplify batch processing with shuffling to enhance diversity in each batch. The mini-batch size is set to 64 to optimize training effectiveness. Model predictions are saved to a excel file for each time step for further comparison with true steering angles.

D. Training

We begin training with the Adam optimizer, setting a weight decay of 1×10^{-5} for regularization, and use mean square error as the loss function to effectively minimize the error between predicted and true steering angles. A scheduler is added to gradually reduce the learning rate as we near the optimal solution, starting from a small initial rate of 1×10^{-5}. To focus learning on steering angle predictions, we freeze all layers in VGGNet except the classifier. For initial model validation, we use the first 20% of each of the five datasets, allowing us to quickly assess the model's setup before committing to full-scale training but not lossing the diversity of dataset.

The first result shows a significant discrepancy from the true steering angle, as evidenced by the plot indicating that nearly all predictions are close to zero. The calculated loss of 0.25 highlights this issue; (the contest guidelines suggest that a reasonable loss should be below 0.20, as 0.20 represents predicting zero for every steering angle, essentially failing to capture any meaningful prediction). This underperformance likely stems from training with shuffled data across both the convolutional and recurrent neural networks. Shuffling is necessary for the convolutional network to ensure exposure to varied road conditions, preventing overfitting to specific scenarios. However, when we then feed these shuffled outputs into the recurrent neural network, it disrupts the sequential order needed to understand temporal relationships, causing it to produce low, unrepresentative predictions. Using unshuffled data across both networks isn't viable either, as it would lead the convolutional network to overfit to particular road situations.

To manage the model's complexity and differing requirements between the networks, we divided the training into three stages. First, we temporarily disabled the recurrent neural network components and trained only the VGGNet with only classifier unfroze using a shuffled dataset to verify that it alone could accurately predict steering angles; the results (detailed in the Results section) were promising. Next, we froze the VGGNet and trained only the LSTM on an unshuffled dataset, confirming that by incorporating 10 previous time steps, the LSTM could reliably predict the subsequent steering angle. Following this, we integrated and trained the Kalman filter structure using the unshuffled dataset. With each component validated, we finally unfroze every layer including all layers of the pre-trained VGGNet and trained the entire model using the complete dataset.

IV. Results

To evaluate our results according to the contest's standards, we calculate the mean square error between the model's predictions and the true steering angles from the test dataset, referring to this metric as the "loss grade." Since we need to maintain the sequence to compare with the ordered true steering angles, shuffling the dataset at this stage is not an option. Our convolutional network alone achieved a loss grade of 0.88, positioning us in 7th place among contest participants—a remarkable achievement given our two-week timeline compared to the contest's two-month preparation period. This demonstrates the strength of transfer learning. The graph below shows our predictions, accurately capturing both values and trends.

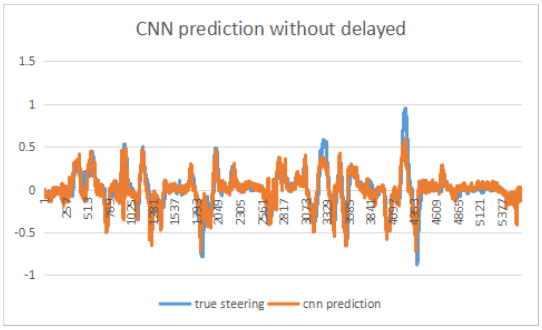


Fig. 3. CNN steering angle prediction without delay

Using the same evaluation method for the entire model, we achieved a loss grade of 0.89, very close to the previous result from the convolutional network alone. This indicates that the Kalman filter and system model identification are still effectively capturing the steering angle, validating the feasibility of our approach. The graph below illustrates the model's performance, accurately reflecting both the values and trends in steering angle predictions.

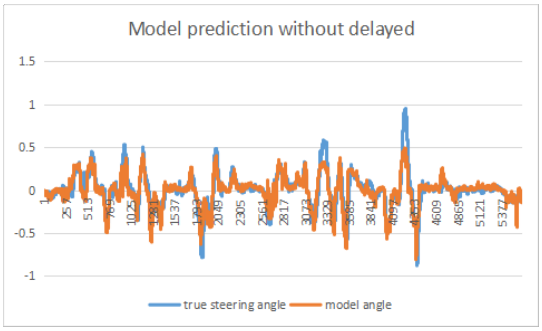
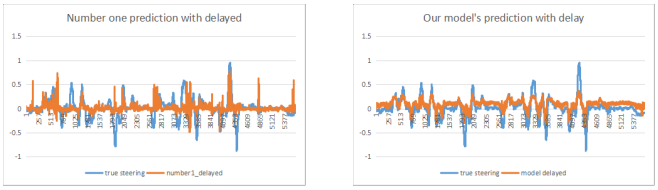


Fig. 4. Full model steering angle prediction without delay

To test the robustness of our model, we introduced a realistic delay scenario by simulating image processing lag—sending an image at every alternate time step (e.g., first image, then a blank, then the third image, etc.). This mirrors real-world conditions where car computers may face processing delays, making timely image data at every step unlikely. We compared our model's predictions to those of the contest's top competitor. Our model's loss grade, though increased to 0.14, still outperformed the top competitor's result, which saw a more significant loss rise to 0.20. This outcome highlights our model's resilience, largely due to the LSTMs effectively managing temporal dependencies, thus mitigating delay effects as we predicted. The following graphs illustrate this performance, where the blue line represents the true steering angle, and the orange line shows the predictions from each model.



(a) Number one prediction with delay.

(b) Our model prediction with delay.

Fig. 5. Comparison between our model and the number one performance in the contest with delayed images

V. Conclusions

In summary, we balanced traditional autonomous driving approaches with machine learning techniques, creating a hybrid method to predict steering angles using front camera images for autonomous vehicles, and the result is tested to be quite well. The network comprises a VGGNet19-based convolutional neural network for visual input, alongside three LSTM networks that predict state and measurement covariance matrices and the system model. The model was trained in Python, initially using one-fifth of the dataset in three steps: shuffled data for VGGNet19, unshuffled data for the system model LSTM, and then with Kalman-filter-structured LSTMs. After achieving promising results, we trained the entire model on the full dataset. The model performed well, achieving a loss score of around 0.88, placing it 7th on the contest leaderboard. Remarkably, when tested with delayed images, our model outperformed the top-ranked competitor by maintaining accurate steering angle predictions.

While we've made significant progress in this project, there is still room for improvement. Our model isn't yet fully optimized, as we've only reached 7th place rather than the top position. The training process is also quite complex, and with more time, we would explore additional approaches for comparison, potentially finding more effective ways to handle time-dependent predictions, address delayed images, or improve training techniques. Moreover, our results are currently only validated on the contest's test dataset. To evaluate the model's generalization further, we would need to test it with data from other sources or, ideally, conduct trials on a real vehicle to ensure robustness.

References

[1] Jelena Kocić, Nenad Jovičić, and Vujo Drndarević. An end-to-end deep neural network for autonomous driving designed for embedded automotive platforms. *Sensors*, 19(9), 2019.

[2] Jonas Heylen, Seppe Iven, Bert De Brabandere, Jose Oramas M., Luc Van Gool, and Tinne Tuytelaars. From pixels to actions: Learning to drive a car with deep neural networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 606–615, 2018.

[3] Udacity. Teaching a machine to steer a car, 2016.

[4] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars, 2016.

[5] Hoang Tran Ngoc, Phuc Phan Hong, Anh Nguyen Quoc, and Luyl-Da Quach. Steering angle prediction for autonomous vehicles using deep transfer learning. *Journal of Advances in Information Technology*, 15(1):138–146, 2024.

[6] Guy Revach, Nir Shlezinger, Xiaoyong Ni, Adria Lopez Escoriza, Ruud J. G. van Sloun, and Yonina C. Eldar. Kalmannet: Neural network aided kalman filtering for partially known dynamics. *IEEE Transactions on Signal Processing*, 70:1532–1547, 2022.

[7] Huseyin Coskun, Felix Achilles, Robert DiPietro, Nassir Navab, and Federico Tombari. Long short-term memory kalman filters: Recurrent neural estimators for pose regularization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5524–5532, 2017.