Optimizing CNNs for Real-Time Autonomous Driving on AutoPicar

Shayan Shafquat[1]

[1]*School of Physics and Astronomy, University of Nottingham, Nottingham, NG7 2RD, UK*

(Dated: July 28, 2024)

The progress in Convolutional Neural Networks (CNNs) have significantly enhanced the capabilities of autonomous driving systems. This paper focuses on optimizing CNNs for real-time autonomous driving using the AutoPicar testbed. By leveraging transfer learning with pre-trained models, a deep learning control system for steering and speed prediction was developed. The approach integrates data augmentation, fine-tuning, hyperparameter tuning, and exploring pooling layers to enhance model performance. Evaluations using Kaggle test data revealed that ResNet50V2 excels in lane navigation, while VGG-16 is superior for obstacle and pedestrian detection. Real-time track testing demonstrated that MobileNetV2 offers a good balance between performance and lower computational demands. This study underscores the importance of quality training data, model selection and optimization techniques in developing robust and efficient CNN-based systems for autonomous driving, achieving the crucial balance between accuracy and real-time responsiveness necessary for practical applications.

I. INTRODUCTION

Advancements in artificial intelligence (AI), machine learning (ML), and deep learning (DL) have greatly enhanced the capabilities of autonomous vehicles. These technologies are set to revolutionize transportation by integrating sophisticated visual recognition systems (VRS) for tasks such as image classification, object detection, segmentation, and localization, which are crucial for effective visual processing [1, 2]. Autonomous vehicles require advanced perception and cognition to handle complex driving situations and ensure safe, reliable manoeuvring [3].

The evolution of deep neural networks (DNNs) has been pivotal in autonomous vehicle control. Projects like DARPA's DAVE and NVIDIA's DAVE-2 demonstrated how convolutional neural networks (CNNs) can interpret visual data directly into steering commands, bypassing traditional control algorithms [4, 5, 6]. This shift highlights the growing reliance on real-time, neural network-based control systems in automotive robotics. Implementing these AI technologies in vehicle-embedded systems presents unique challenges as the inference phase requires efficient processing to meet stringent real-time operational demands essential for vehicular safety [7, 8].

This study introduces AutoPicar as depicted in Figure (1), which leverages an end-to-end deep learning-based control system using a CNN. Utilizing transfer learning, CNN employs pre-trained models like MobileNetV2, ResNet50V2, and VGG-16, optimized for autonomous navigation challenges. The network processes images from a forward-looking camera to make real-time steering and speed decisions.

This research evaluates AutoPicar's performance on Kaggle test data and its real-time capabilities on the Coral Edge TPU across various scenarios. The study focuses on different CNN architectures, assessing the impact of data augmentation, fine-tuning, and hyperparameter tuning on model effectiveness. Architectural modifications such as pooling layers, optimizers, and loss functions are also explored.

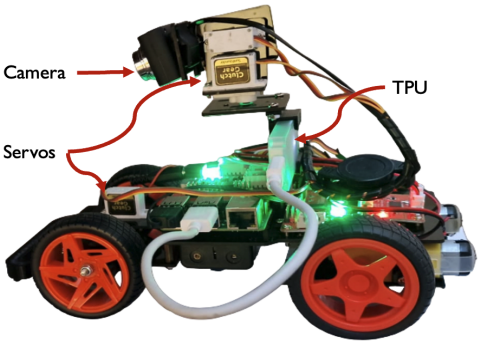

FIG. 1: **SunFounder Picar.** The autonomous robotic vehicle used in the project is equipped with a camera, a Tensor Processing Unit (TPU), and servos. The camera captures real-time data for image recognition tasks, the TPU processes machine-learning algorithms efficiently, and the servos facilitate precise movement control.

The aim is to address key questions about CNN-based architecture performance, the role of enhancement techniques in model accuracy, and the balance between accuracy and inference speed in real scenarios. This includes optimizing CNNs for real-time processing to ensure high performance and responsiveness in practical autonomous driving applications.

II. METHODOLOGY

A. Data Preparation and Augmentation

Exploratory data analysis revealed that the training set comprised over 13,000 images with corresponding vehicle responses in terms of speed and steering angles. The test set included over 1,000 samples. Initial quality

checks identified and corrected mislabeled samples to ensure data accuracy.

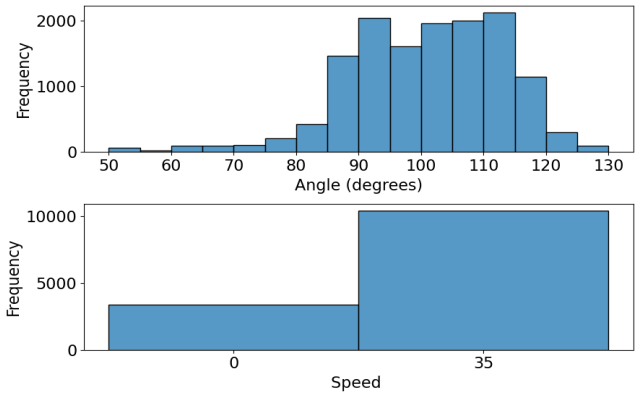

FIG. 2: **Distribution histograms for ground-truth angle and speed values.** The histogram reveals the class imbalance in the training labels for steering angles and speed

Figure (2) shows significant class imbalances, particularly with extreme steering angles and binary speed values. To mitigate this and enhance robustness, data augmentation techniques were implemented. These methods are pivotal in preventing model overfitting by enriching the diversity of the training data.

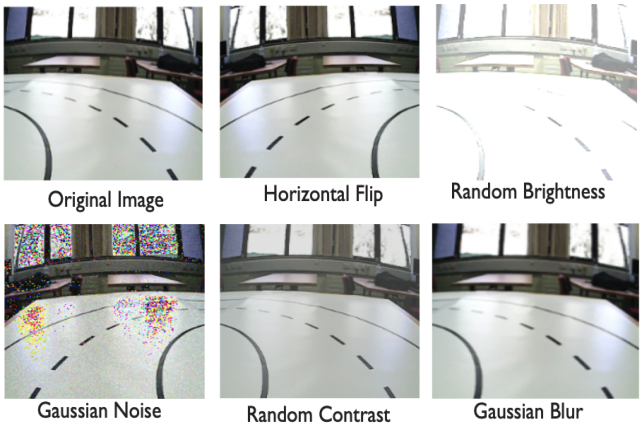

FIG. 3: **Image Transformations for Data Augmentation.** Displayed are various augmentations applied to a baseline image captured by the vehicle: Original Image, Horizontal Flip, Random Brightness Adjustment, Gaussian Noise, Random Contrast, and Gaussian Blur.

As depicted in Figure (3), a range of image transformations, such as horizontal flips and adjustments to brightness, contrast, and sharpness, were utilized. These augmentations enable the model to interpret various visual scenarios accurately, ensuring reliable performance across different lighting and environmental conditions.

B. Model Development

The model development strategy incorporated deep learning models to enhance performance and accuracy for tasks such as angle and speed prediction. This approach leveraged distinct architectures and transfer learning techniques to tailor model's intended function.

Angle prediction is treated as a multi-classification problem, utilizing one-hot encoding and class weights in the loss function to mitigate class imbalance effects. Conversely, speed prediction is approached as a binary classification task, employing one-hot encoding but focusing on different aspects of model training to optimize speed-related decisions.

1. *Transfer Learning*

Transfer learning was a key strategy to bolster model performance without extensive data collection and training from scratch. Using a pre-trained Convolutional Neural Network (CNN) on ImageNet [9], models learned generic features transferable across various visual tasks [10]. Fine-tuning adjusted the final layers to specific requirements—angle and speed prediction—ensuring the models were well-adapted to the project's nuances (see Figure 4).

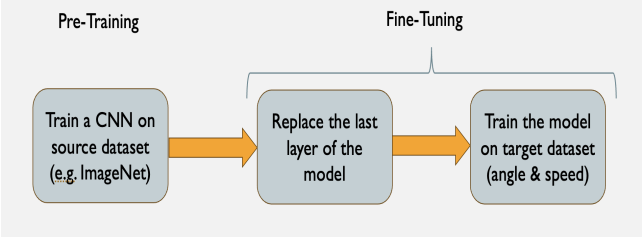

FIG. 4: **Transfer Learning Process.** Outline of the transfer learning steps.

2. *Model Architectures*

The effectiveness of the predictive models hinges significantly on architectural choices. A variety of base models were employed, each selected based on its strengths and suitability for the specific computational demands of the tasks (see Figure 5).

Each architecture begins with a robust input layer feeding into one of several base models: VGG-16, known for its depth and high parameter count [11]; ResNet50V2, which introduces residual learning to aid deeper network training [12]; and MobileNetV2, optimized for efficiency in resource-constrained environments [13]. These base models, pre-trained on ImageNet [9], provide diverse features crucial for complex image recognition tasks in autonomous driving [14].

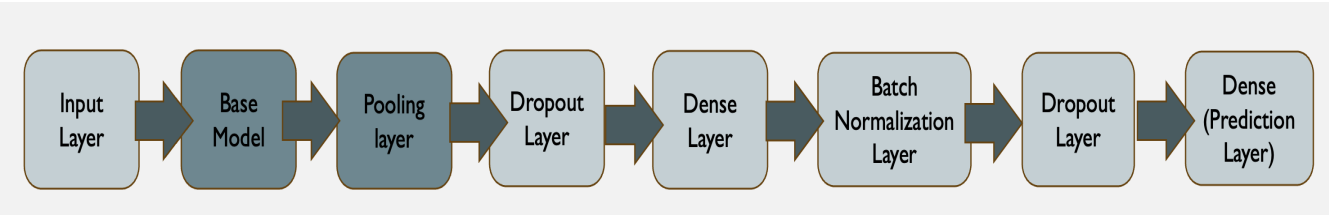

FIG. 5: **Neural Network Architecture Utilizing Transfer Learning.** The diagram showcases the neural network setup for training AutoPicar's model.

Following the base model, the architecture includes pooling layers to reduce dimensionality, dropout layers to combat overfitting, and dense layers to facilitate learning complex feature combinations. Batch normalization layers ensure stable learning across various driving scenarios.

Integrating advanced architectures and transfer learning enhances the models' ability to interpret complex driving environments, ensuring robust performance in real-world applications.

3. *Pooling Layers*

The analysis focuses on the strategically implementing of pooling layers within convolutional neural networks (CNNs), optimizing the architecture for specific image recognition tasks in autonomous driving. The choice of pooling layer—

GlobalAveragePooling2D aggregates spatial information efficiently, averaging out each feature map. This reduces model complexity and helps prevent overfitting, which is crucial for tasks requiring robust generalization across spatial variations.

GlobalMaxPooling2D extracts the most significant features by capturing the maximum value from each feature map. This technique is valuable when distinguishing prominent features is essential, such as detecting specific obstacles or road signs.

SpatialPyramidPooling2D (SPP) which is inspired by the architectural enhancements in [15], handles inputs of varying sizes, creating a fixed-length output independent of input size by implementing multi-level pooling. Our implementation defined SPP using a custom TensorFlow layer, which performed max pooling over different region sizes specified in the pool list [1,2,4]. This approach allowed the model to aggregate features at multiple scales, enhancing its ability to capture spatial information across diverse input sizes [15].

The selection should be guided by the specific demands of the dataset and the task at hand. For example, while average pooling is beneficial for noise reduction in classification, max pooling may be preferable for tasks prioritizing feature presence [16].

4. *Loss Functions and Optimization Strategies*

Specialized loss functions are implemented to address class imbalance to enhance sensitivity towards underrepresented classes. Categorical Focal Cross-Entropy is used for angle prediction, and Binary Focal Cross-Entropy for speed prediction. These focal loss functions modify traditional cross-entropy loss to concentrate more on difficult-to-classify instances, reducing the dominance of majority classes [17] (see Equation 1).

$$FL(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t) \tag{1}$$

where p_t represents the predicted probability of the true class, α_t denotes the modulating factor that adjusts the learning rate of each class, and γ is a tunable focusing parameter that diminishes the contribution to the loss for well-classified examples. This approach prioritizes misclassified or hard-to-be-classified examples over those already well-classified, enhancing model accuracy and robustness across diverse driving conditions [17].

Alongside these advanced loss functions, optimization algorithms such as RMSProp and ADAM are utilized to fine-tune the models. RMSProp stabilizes training updates by normalizing the gradients using a moving average of their squared values. ADAM, an enhancement of RMSProp, incorporates momentum to improve convergence, allowing adaptive adjustments of learning rates for each parameter based on historical gradient data [18].

Employing these strategies ensures precise updates during training, leading to improved model performance and generalization capabilities.

5. *Evaluation Metrics*

Evaluating machine learning models is essential for assessing performance and suitability for real-world applications. This study employs various evaluation metrics to comprehensively analyze the model's effectiveness.

Mean Squared Error (MSE) is primarily utilized for regression tasks to quantify the average of the squares of the errors—that is, the average squared difference between the predicted and actual values. MSE is particu-

larly relevant as the predicted classes, namely steering angle and speed, are numerical.

For classification tasks, while accuracy is a standard measure of performance, it can be misleading in imbalanced datasets. Therefore, this study emphasizes the importance of the Confusion Matrix as a more informative metric. The Confusion Matrix reveals the number of correct and incorrect predictions and categorizes these by class, providing deeper insight into the model's performance across different categories.

C. Model Deployment

In the deployment phase, the MobileNetV2 model [13], known for its efficiency on mobile platforms, was refined to enhance performance across various real-world conditions, detailed in Figure (6). To address training data for underrepresented driving scenarios, the dataset was augmented with over 1,000 additional images through data collection.

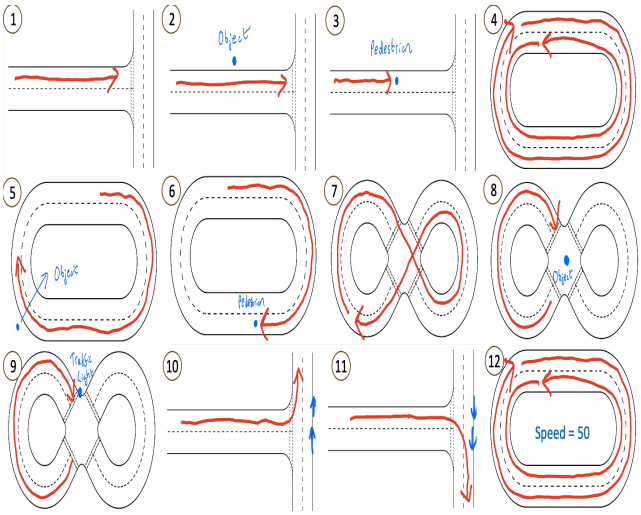

FIG. 6: **Real-Time Testing Scenarios for Autonomous Vehicle Navigation.** The twelve challenges include obstacle avoidance, pedestrian detection, traffic signal compliance, and handling intersections, demonstrating the comprehensive testing environment.

To further optimize the model for mobile deployment, the input image size was reduced from 192x192 to 128x128 pixels, decreasing computational demands and enhancing processing speed—vital for the real-time responsiveness required in autonomous driving. The model was then converted into TensorFlow Lite format using the FlatBuffer approach, resulting in a compact, efficient .tflite file. This conversion ensured the model remained lightweight and performed efficiently in low-resource settings. Maintaining operational compatibility and minimizing inference times ensured the model's practical applicability and reliable performance without compromising accuracy.

III. RESULTS

A. Comparison of Base Models

In the comparison of convolutional neural network models for an autonomous driving car project, the VGG-16 model exhibited superior performance in terms of validation loss and mean squared error (MSE) on the speed prediction task, achieving the lowest validation loss and MSE values of 0.0096 and 0.0120, respectively, as indicated in (Figure 7) and (Table I). Conversely, the ResNet50V2 model was the most effective for the angle prediction task, demonstrating the lowest validation loss and MSE values of 0.2360 and 0.0414, respectively (see Table I). All models were trained using an input image size of 160x160 and a batch size of 128 with finetuning the final 25 layers of base model in the last 20 epochs as in (Figure 7).

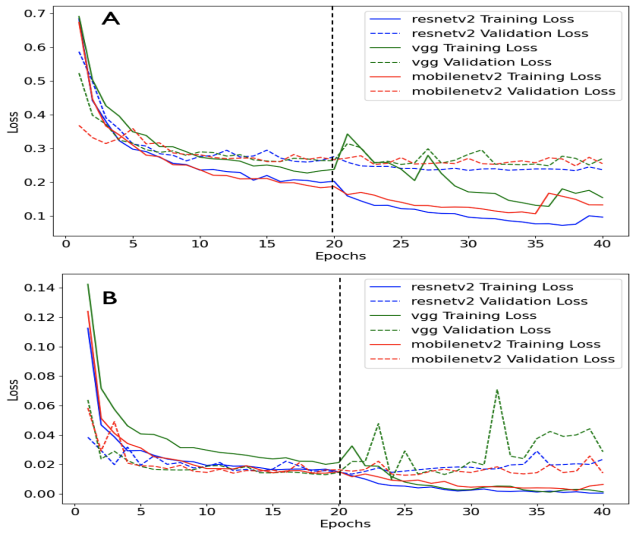

FIG. 7: **Training and validation loss comparison across CNN models.** Subfigures A and B display the training and validation loss curves for angle and speed prediction tasks over 20 epochs of initial training and 20 epochs of finetuning. Solid lines indicate training loss, and dashed lines represent validation loss.

Manoeuvring	Model	Train Loss	Val Loss	Val MSE	Test MSE
Angle	**ResNet50V2**	0.1108	**0.2360**	**0.0414**	
Speed		0.0123	0.0135	0.0129	0.0225
Angle	**VGG-16**	0.1759	0.2513	0.0435	
Speed		0.0117	**0.0096**	**0.0120**	**0.0151**
Angle	**MobileNetV2**	0.1491	0.2481	0.0429	
Speed		0.0086	0.0127	0.0122	0.0224

TABLE I: Performance metrics of CNN models on training, validation, and test datasets for autonomous driving tasks.

Furthermore, results on the test dataset reveal that the VGG-16 model offered the best results in speed prediction with a test MSE of 0.0151 (refer to

Table I). Despite its lower computational demands, the MobileNetV2 model delivered competitive outcomes, closely approximating the performance of more resource-intensive models. This efficiency is underscored by its MSE results for both speed (0.0122) and angle (0.0429) predictions, suggesting its viability in resource-constrained environments.

This analysis confirms the critical role of model selection in optimizing performance and resource allocation for specific tasks within autonomous vehicle systems.

B. Tuning Hyperparameters, Pooling Layers, and Optimizers

Table (II) details the optimization of hyperparameters, pooling layers, and optimizers for both speed and angle prediction models, highlighting their impact on validation loss.

For the speed models, a configuration using spatial pyramid pooling (SPP) and a learning rate of 0.000275 achieved the lowest validation loss of 0.01459. Other setups, including both SPP and max pooling, displayed validation losses ranging from 0.01460 to 0.01463, showing stable performance despite variations in dropout rates and dense units.

In contrast, the angle models had a broader range of validation losses. The most effective configuration used average pooling and a learning rate of 0.00180, yielding the lowest validation loss of 0.25716. Other configurations with max, SPP, and average pooling resulted in validation losses between 0.25175 and 0.25801, indicating that the choice of optimizer and subtle adjustments in learning rate are crucial for refining model accuracy.

These findings underscore the significant influence of specific hyperparameters, particularly pooling type and learning rate, on the performance of CNN models. This detailed analysis provides critical insights into configurations that optimize performance for tasks essential to autonomous driving, highlighting how targeted adjustments can enhance model validation accuracy.

C. Real-time Testing of AutoPicar

The MobileNetV2-based model was retrained using optimized configurations identified previously, excluding spatial pyramid pooling (SPP) to facilitate conversion to TensorFlow Lite (TFLite). This adaptation aimed to avoid complexities associated with custom layers during the conversion process. Upon evaluation of the Kaggle dataset, the model significantly improved, achieving a mean squared error (MSE) of 0.01390. When deployed on a TPU device, the inference time averaged 120 milliseconds per manoeuvre. However, conversion to TFLite reduced the inference time to 20 milliseconds, slightly increasing in MSE to 0.01605.

In real-world testing on the PiCar, the model completed 8 out of 12 challenges, demonstrating substantial capability in practical scenarios. It performed well at a speed setting of 50, corresponding to scenario 12. Challenges were noted in scenarios involving more complex navigational tasks: it failed in scenario 4 (inner track manoeuvring), scenario 7 (navigating straight at a junction), scenario 9 (stopping at a traffic light), and scenario 10 (executing a left turn).

The confusion matrices further elucidate the model's operational capabilities and limitations. The angle model's confusion matrix (Figure 8) shows several off-diagonal entries, indicating mispredictions in lower angle values, contributing to difficulties in manoeuvring towards the left. Similarly, the speed model's confusion matrix (Figure 9) highlights misclassifications affecting performance in speed prediction tasks like failing to interpret green traffic light signals on different tracks.

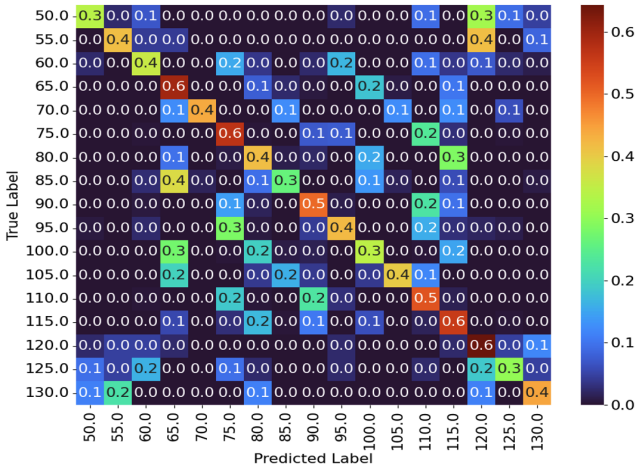

FIG. 8: **Confusion Matrix for Angle Prediction Model**. This matrix displays the model's prediction accuracy across various angle settings, with notable off-diagonal entries indicating mispredictions.

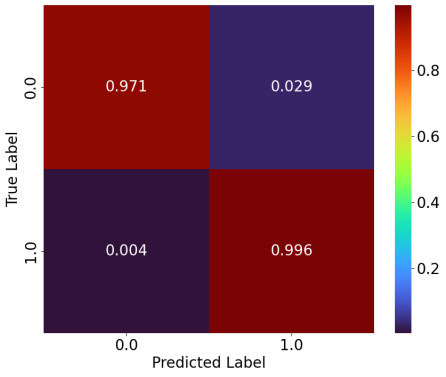

FIG. 9: **Confusion Matrix for Speed Prediction Model**. This matrix shows the prediction accuracy for speed settings, highlighting areas where the model misclassified speed values.

Speed Model Tuning Results

Val Loss	Brightness	Contrast	Pooling Type	Dropout	Dense Units	Final Dropout	Optimizer	Learning Rate
0.01459	0.05	0.1	spp	0.3	96	0.3	rmsprop	0.00275
0.01460	0.25	0.1	spp	0.3	224	0.2	rmsprop	0.00125
0.01461	0.2	0.25	max	0.4	352	0.3	adam	0.00385
0.01462	0.2	0.2	max	0.3	224	0.3	adam	0.00301
0.01463	0.2	0.05	max	0.4	416	0.2	adam	0.00111

Angle Model Tuning Results

Val Loss	Brightness	Contrast	Pooling Type	Dropout	Dense Units	Final Dropout	Optimizer	Learning Rate
0.24515	0.2	0.1	spp	0.3	256	0.3	rmsprop	0.00241
0.25603	0.2	0.2	max	0.2	256	0.2	adam	0.00219
0.25716	0.2	0.2	avg	0.2	256	0.2	adam	0.00251
0.25750	0.2	0.1	spp	0.3	1024	0.4	rmsprop	0.00177
0.25801	0.2	0.1	avg	0.3	512	0.2	adam	0.00180

TABLE II: Tuning Results Highlighting the Optimization of Hyperparameters, Pooling Layers, and Optimizers for Speed and Angle Models

Overall, these results highlight model's effectiveness in real-life scenarios as in (Figure 6) while pointing to situations requiring further enhancement to ensure better performance across all scenarios.

IV. DISCUSSION

Angle Prediction. The angle prediction problem was tackled as a multi-classification task rather than a regression task, allowing for the assignment of class weights to address class imbalances effectively. However, regression seems a more plausible option to maintain ordinal relationships among the angles, where tackling class imbalance becomes more challenging and hence avoided.

Custom Layers. We experimented with custom layers such as GaussianBlur and SpatialPyramidPooling [15] in TensorFlow 2.15, installed in our vehicle's computing system. While these layers could enhance model performance, their complexity hindered conversion to TensorFlow Lite (TFLite), leading to their omission in the final deployment. Although horizontal flip transformations improved results on the Kaggle test dataset, they were excluded from deployment to ensure adherence to traffic regulations and avoid non-realistic manoeuvres.

Undersampled Classes. The training dataset had underrepresented samples for scenarios, like for left turns on straight tracks, leading to poor vehicle performance in these situations. This highlights the importance of comprehensive and representative data collection for training deep learning models in autonomous driving. Additional data collection for scenarios like traffic light signals and left turns improved model performance in subsequent Kaggle evaluations. However, converting to TFLite diminished performance compared to the original model, underscoring the trade-off between model complexity and deployment efficiency.

This discussion highlights the multifaceted challenges in developing the AutoPicar model.

V. CONCLUSION

This study demonstrates the effective optimization of Convolutional Neural Networks (CNNs) for real-time autonomous driving using the AutoPicar testbed. Utilizing transfer learning with models like MobileNetV2, ResNet50V2, and VGG-16 and incorporating data augmentation, fine-tuning, and hyperparameter tuning, led to significant performance improvements. ResNet50V2 excelled in lane navigation, VGG-16 proved effective for obstacle and pedestrian detection, and MobileNetV2 provided a balanced solution for resource-constrained environments.

Exploring different pooling layers, including GlobalAveragePooling2D, GlobalMaxPooling2D, and SpatialPyramidPooling2D, yielded valuable insights into enhancing model effectiveness. The practical challenges of implementing custom TensorFlow layers and addressing class imbalances were effectively managed, ensuring efficient deployment on edge devices.

Real-time testing of the AutoPicar on tracks demonstrated the models' applicability in practical scenarios, though further refinement is needed for complex manoeuvres. This work highlights the critical importance of quality training data, model selection and optimization in developing robust, responsive CNN-based systems for autonomous driving, ensuring a essential balance between accuracy and real-time performance for practical deployment.

VI. ACKNOWLEDGEMENTS

I want to express gratitude to Adam Moss and Maggie Lieu for consistent guidance throughout this project.

REFERENCES

1. Pakusch C, Stevens G, Boden A, and Bossauer P. Unintended Effects of Autonomous Driving: A study on mobility Preferences in the Future. Sustainability 2018 Jul; 10:2404. DOI: 10.3390/su10072404. Available from: https://www.mdpi.com/2071-1050/10/7/2404

2. Trotta G. Computer vision and deep learning techniques for pedestrian detection and tracking: A survey. www.academia.edu 2018 Jan. Available from: https://www.academia.edu/97687409/Computer_vision_and_deep_learning_techniques_for_pedestrian_detection_and_tracking_A_survey

3. Calvert SC, Schakel WJ, and Van Lint JWC. Will automated vehicles negatively impact traffic flow? Journal of advanced transportation 2017 Jan; 2017:1–17. DOI: 10.1155/2017/3082781. Available from: https://www.hindawi.com/journals/jat/2017/3082781/

4. Lecun Y, Cosatto E, Ben J, Muller U, and Flepp B. DAVE: Autonomous Off-Road Vehicle Control Using End-to-End Learning. Tech. rep. DARPA-IPTO Final Report. http://www.cs.nyu.edu/ỹann/research/dave/index.html: Courant Institute/CBLL, 2004

5. Pomerleau DA and Computer Science Department CMU. ALVINN: an autonomous land vehicle in a neural network. Tech. rep. Available from: https://proceedings.neurips.cc/paper/1988/file/812b4ba287f5ee0bc9d43bbf5bbe87fb-Paper.pdf

6. Bojarski M, Davide DT, Dworakowski D, Firner B, Flepp B, Goyal P, Jackel LD, Monfort M, Muller U, Zhang J, Zhang X, Zhao J, and Zieba K. End to end learning for Self-Driving cars. 2016 Apr. Available from: https://arxiv.org/abs/1604.07316

7. GPU-Based Deep Learning Inference: A performance and power analysis. 2015. Available from: https://www.semanticscholar.org/paper/GPU-Based-Deep-Learning-Inference%3A-A-Performance/eabf117aae614d6b65cc33a3a4bbb22b9a5014d9

8. An evaluation of the NVIDIA TX1 for supporting Real-Time Computer-Vision workloads. 2017 Apr. Available from: https://ieeexplore.ieee.org/document/7939053

9. ImageNet: A large-scale hierarchical image database. 2009 Jun. Available from: https://ieeexplore.ieee.org/document/5206848

10. Huh M, Agrawal P, and Efros AA. What makes ImageNet good for transfer learning? 2016 Aug. Available from: https://arxiv.org/abs/1608.08614

11. Simonyan K and Zisserman A. Very deep convolutional networks for Large-Scale image recognition. 2014 Sep. Available from: https://arxiv.org/abs/1409.1556

12. Deep residual learning for image recognition. 2016 Jun. Available from: https://ieeexplore.ieee.org/document/7780459

13. MobileNetV2: Inverted residuals and linear bottlenecks. 2018 Jun. Available from: https://ieeexplore.ieee.org/document/8578572

14. CNN features Off-the-Shelf: an astounding baseline for recognition. 2014 Jun. Available from: https://ieeexplore.ieee.org/document/6910029

15. He K, Zhang X, Ren S, and Sun J. Spatial pyramid pooling in deep convolutional networks for visual recognition. 2014 Jan :346–61. DOI: 10.1007/978-3-319-10578-9\{_}23. Available from: https://doi.org/10.1007/978-3-319-10578-9_23

16. Boureau YL, Ponce J, and LeCun Y. A Theoretical Analysis of Feature Pooling in Visual Recognition. *International Conference on Machine Learning*. 2010. Available from: https://api.semanticscholar.org/CorpusID:2167514

17. Lin TY, Goyal P, Girshick RB, He K, and Dollár P. Focal Loss for Dense Object Detection. 2017 IEEE International Conference on Computer Vision (ICCV) 2017 :2999–3007. Available from: https://api.semanticscholar.org/CorpusID:47252984

18. Kingma DP and Ba J. Adam: A method for stochastic optimization. 2014 Dec. Available from: https://arxiv.org/abs/1412.6980