Second Homework Report on Driving Racing Car

Giuseppe Misuraca, 1944376
misuraca.1944376@studenti.uniroma1.it

Academic year 2024-25

University of Rome "La Sapienza"

Abstract

This report delineates the efforts employed to accomplish a machine learning classification task. The task involved the classification of various actions a car could perform, given as input an image of the vehicle engaged in a track circuit. To accomplish this objective, a range of techniques was employed, resulting in the creation of two distinct models. These models were then subjected to a comparative analysis to assess their efficacy in addressing the task.

1 Dataset Description & Processing

The dataset, which consists of images with a shape of (96, 96, 3), was pre-split into a training subset and a testing subset, comprising 6369 and 2749 files, respectively. The images are frames extracted from a simulation conducted within the Box2D-based environment known as Gymnasium's CarRacing environment. The dataset includes five possible labels, represented as integers, corresponding to the potential actions the car can perform while navigating the circuit. Specifically, these actions are: do nothing, steer left, steer right, accelerate and brake.

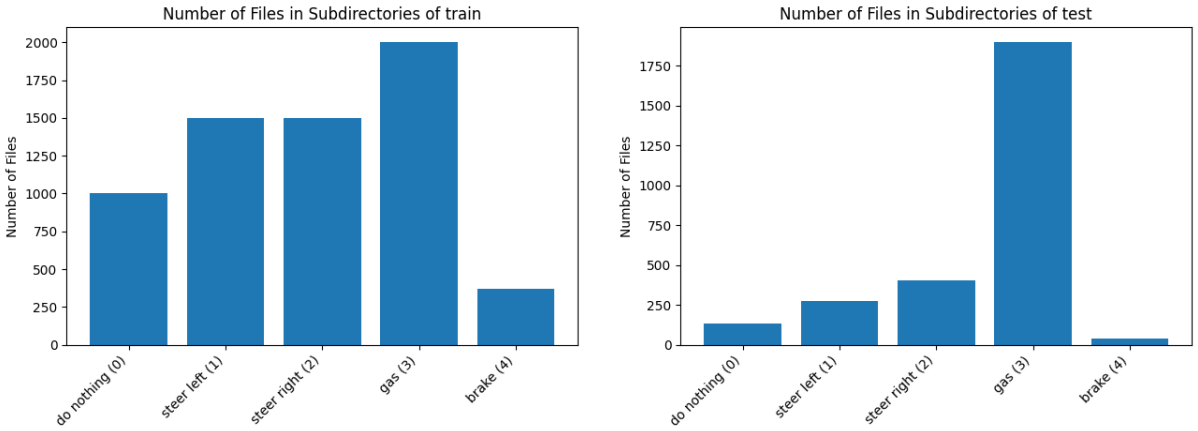

Figure 1: Respectively data distribution of training and testing set.

As illustrated by the histograms, the dataset displays a highly imbalanced distribution, with considerable disparities in the number of images assigned to each label. In the testing set, consisting of 2749 images, the number of files per label varies from 39 to 1896, with the majority of images belonging to label 3 (1896 files, representing 68% of the test set), while label 4 has only 39 images (1.41%). Similarly, the training set also exhibits an uneven distribution, with label 3 comprising the largest portion of the data. This imbalance introduced challenges to the model training process, as the model developed a bias towards the more frequent classes. Consequently, the model demonstrated higher performance in detecting the more frequent labels, while the score on the less frequent labels was suboptimal. To address this issue, the selection of appropriate metrics, such as weighted F1 score, was crucial to ensure a more balanced evaluation of the model's performance across all classes.

The dataset for this classification task presents a second significant challenge due to the presence of samples with overlapping features that are labeled differently. As illustrated in the provided image, several classes correspond to visually similar samples with subtle differences, such as minor variations. This overlap introduces noise into the dataset, complicating the model's ability to learn discriminative features specific to each class.

One approach considered was to remove noisy samples to address this issue. However, this strategy proved counterproductive. Eliminating noisy or ambiguous samples often resulted in the removal of entire classes or a substantial portion of the dataset, leaving too little data to train a robust model effectively. Moreover, such reductions in data size and diversity risked producing a dataset that was no longer representative of real-world scenarios, where visual ambiguity and overlapping features are commonplace. As a result, the decision was made to retain the noisy samples, acknowledging the inherent challenge and aiming to develop a model that can generalize effectively despite these complexities. As expected, this issue manifested prominently in the analysis of the confusion matrix. Due to the presence of overlapping features and the existence of noisy labels, a certain number of samples have been misclassified.

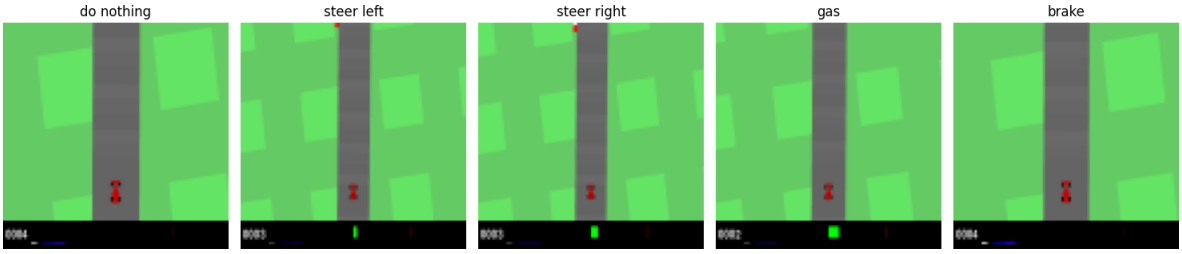

Figure 2: Examples of Noisy Labels in the Dataset.

1.1 Data Processing

The images supplied to both models underwent a preprocessing phase prior to being utilized as input. This preprocessing step was uniformly applied to ensure comparability between the models, thereby enabling an assessment of whether different models, when trained on identical data, exhibit significant performance variations.

The initial step in data preparation involved partitioning the dataset into training, validation and testing subsets. As previously mentioned, the training and testing samples were pre-split; thus, the remaining task was to separate the training data into training and validation sets. For this purpose, 80% of the training set was allocated for model fitting, while the remaining 20% was reserved for validation to monitor and evaluate the model's performance during training. The next step involved converting the integer-encoded labels into a one-hot encoding format.

Subsequent image processing steps entailed the normalization and resizing of the images. Normalization is a common preprocessing step in machine learning as it helps to standardize the input data, making the training process more stable and efficient. It ensures that the model's weights are updated more uniformly, leading to faster convergence during training. Then the pictures were resized to ensure that all input images had the same dimensions, which is a requirement for the models used.

During the initial stages of task development, there was uncertainty regarding the application of image transformations prior to feeding them into the model. However, a decision was made to bypass such transformations for the following reasons:

- **Image Orientation is Crucial**: For tasks where image orientation is a critical factor, such as in this case, applying transformations like rotation may not be appropriate.
- **Potential Introduction of Bias**: In the context of small datasets, transformations can introduce bias or artifacts that do not exist in real-world data. Specifically, for this task, where the images were captured in a controlled environment with consistent properties such as color, orientation and other characteristics, altering these attributes could result in a biased model.

A secondary data processing approach that was initially considered was the creation of a balanced dataset by equalizing the cardinality of each label. This method involved reducing the frequency of overrepresented labels to match that of the least frequent label. However, this approach was ultimately deemed suboptimal due to two primary challenges. First, the dataset's label distribution exhibits significant variability. For instance, in the testing set, the most frequent label contains 1896 samples, while the least frequent label has only 39 samples. To achieve balance, all labels would need to be reduced to 39 samples each. Such a drastic reduction in data would severely diminish the size of the dataset, compromising the model's ability to generalize effectively and leading to a substantial loss of information. The second reason was that the dataset reflects the real-world simulation environment and the actual distribution of choices made during the simulation. Artificially altering the label frequencies would distort this natural distribution, resulting in a dataset that no longer accurately represents the problem domain. This misrepresentation could negatively impact the model's performance when deployed in real-world or simulated environments, as the learned patterns would no longer align with the actual data distribution.

Given these considerations, the decision was made to preserve the original label frequencies and address class imbalance through alternative methods, such as weighted metrics or data augmentation strategies, to ensure the model remains representative of the real-world scenario while effectively learning from the available data.

1.2 Data Augmentation

During experimentation, an alternative approach was explored to enhance the model's performance by training it on a larger dataset. To achieve this, a standard machine learning technique known as data augmentation was incorporated into the pipeline. This technique generates synthetic data by applying various transformations to the original images, ensuring that the augmented data remains representative of the problem domain. By introducing slight variations to the original images, data augmentation effectively increases the diversity of the training set, enabling the model to learn more robust features and improving its overall performance. Despite the inherent risks associated with data augmentation for orientation-dependent tasks and the potential for introducing bias in small datasets, a limited set of transformations were carefully selected to enhance model generalization without significantly altering crucial image characteristics.

The following is a detailed list of the transformations and their corresponding reasoning:

- **Random Rotation**: This transformation introduces random rotations to the training images. With a small factor of 0.1, the rotations applied are minimal, reducing the likelihood of significantly impacting the model's ability to discern orientation-related features.
- **Random Zoom**: This augmentation applies minor random zooms to the images. The small value of 0.01 guarantees that the zoom effect is kept to a minimum, preserving the overall scale and context of the objects within the images. The goal is to introduce slight variations in object size, enhancing the model's ability to handle minor scale changes.
- **Gaussian Noise**: This transformation adds Gaussian noise to the images. With a standard deviation of 0.2, the noise is moderate, preventing excessive degradation of image quality. By introducing noise, the model is encouraged to learn more robust features, improving its resilience to noise present in unseen data.

The generation of synthetic data through the aforementioned transformations was executed in a manner that resulted in a twofold increase in the number of samples in the training set, so for each sample, the set of transformations was applied. It is important to emphasize that image augmentation was applied exclusively to the training set. This approach is essential because the validation and testing sets serve as unbiased benchmarks for evaluating the model's performance on unseen data. Introducing data augmentation to these sets would artificially alter their characteristics, potentially misrepresenting real-world scenarios. Furthermore, applying augmentation to the validation or testing sets could result in information leakage, as it would indirectly introduce information about these sets into the training process. By limiting augmentation to the training set, the independence of the validation and testing sets is preserved, ensuring a more accurate and realistic assessment of the model's generalization capabilities.

A comparative analysis has been implemented to ascertain the efficacy of implementing a model trained on an original dataset versus a model trained on a dataset merged with synthetic data.

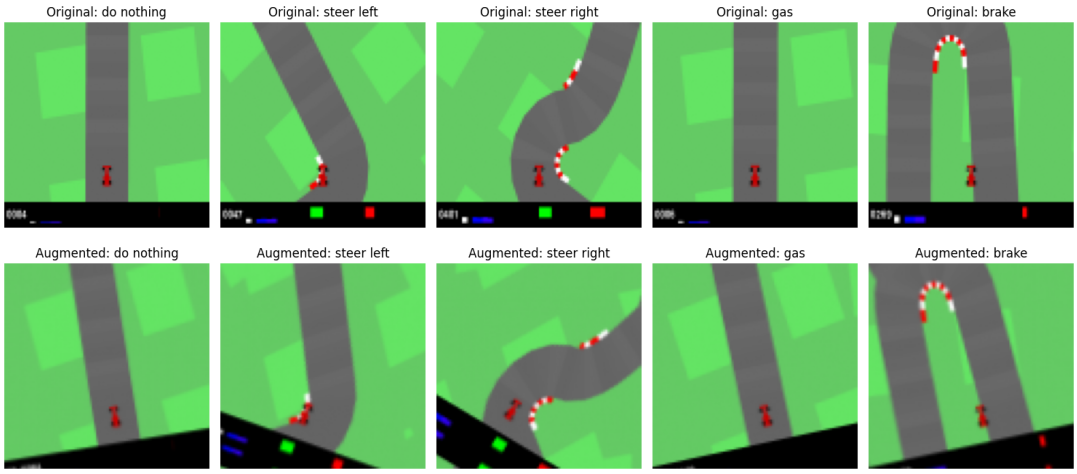

Figure 3: Comparison of Original and Augmented Images.

2 Evaluation Metrics

Given the imbalanced nature of both training and testing sets, it was determined that the weighted f1 metric would be employed for evaluation, as opposed to the accuracy metric. The accuracy metric has the potential to produce misleading results in imbalanced datasets because it may assign a high score to a model that predominantly predicts the majority class, while neglecting to consider the minority class, which, in this case, is particularly relevant.
Although the standard weighted f1 implementation, which is frequently utilized in sklearn's classification reports and calculates weights based on the number of samples (support) per class, aligns with common practices and facilitates comparison with other evaluation systems, it has the capacity to overemphasize the performance on the dominant classes and underrepresented the significance of minority classes in highly skewed distributions.

In order to prioritize a more nuanced assessment that reflects the true distribution of classes and ensures adequate representation of minority classes, a custom weighted f1 implementation was adopted. This approach involves weighting each class according to its actual occurrence in the ground truth (y_true), thereby emphasizing the model's performance on underrepresented classes. This implementation is consistent with the objective of achieving a fair and comprehensive evaluation in scenarios where rare classes are critical, such as in this case where the turning action is not the dominant label.

During the training phase, the model's performance was primarily assessed using the custom-weighted f1 score as evaluation metric, in addition to this, the accuracy of the model was also recorded to provide a broader perspective on its overall performance.
During the testing phase, a detailed evaluation was conducted. This included the generation of a classification report, which provides a breakdown of precision, recall, standard f1 score and support for each class, offering insights into the model's performance on a per-class basis. Additionally, a confusion matrix was plotted to visualize the number of true positives, true negatives, false positives and false negatives for each class.

3 First Model

3.1 Model Description

The initial model employed to address this task is a convolutional neural network (CNN). Prior to the development of the final model, a series of experiments were conducted to ascertain the optimal configuration, resulting in the following architecture, which was deemed most effective. The implemented CNN was designed to classify data through a structured sequence of convolutional, normalization, pooling and dense layers.

The model begins with an input layer that accommodates images with a pre-defined shape, ensuring compatibility with the dataset. The initial convolutional layer is characterized by the application of a series of filters that have a fixed kernel size of 3. These filters are also subject to a variable that controls the stride parameter. The ReLU activation function is used throughout the convolutional layers to introduce non-linearity, a critical component in capturing complex patterns in the data. Each convolutional layer is followed by batch normalization. To progressively reduce the spatial dimensions and focus on salient features, an average pooling layer (2,2) is applied after each convolutional block. The number of filters in each convolutional layer remains consistent throughout the network, resulting in a uniform architecture where the representational capacity of the CNN neither diverges nor converges across layers. This choice ensures that the network maintains a steady level of feature extraction capability at each stage, avoiding abrupt changes in complexity.
Following the last convolutional layers, the feature maps are flattened, transitioning the architecture into its dense layer component. The fully connected dense layers serve to integrate the spatial features into class predictions. The first dense layer is configured with a number of units proportional to the capacity of the network to enhance its representation power. The final dense layer generates predictions for the target classes by employing a softmax activation function, thereby providing probabilities for each class. The network is optimized using the Adam optimizer, which provides robust convergence properties across a variety of tasks and datasets. The loss function implemented is categorical cross entropy, a suitable choice for multi-class classification, as is appropriate for the task at hand.

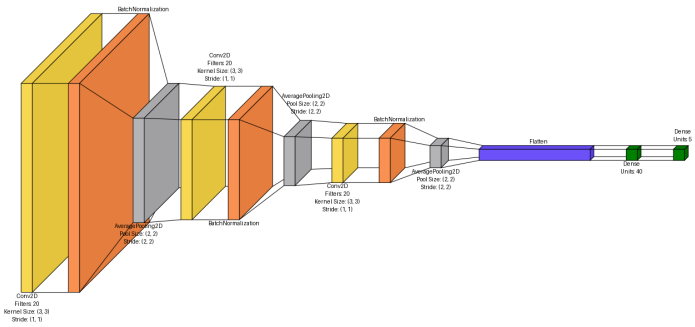

Figure 4: First CNN Model Architecture

As referenced previously, multiple alternative approaches were explored prior to converging on the final model configuration. One of these involved the application of label smoothing in the categorical cross-entropy loss function. Label smoothing was initially hypothesized to improve generalization by mitigating overconfidence in the model's predictions. While this technique demonstrated a modest reduction in overfitting during validation, the overall improvement in predictive performance on the test set was not statistically significant. Another variation involved extending the training process to 100 epochs, significantly longer than the 40 epochs utilized in the final configuration. This approach sought to capitalize on the model's capacity to refine its internal representations over a prolonged training period. However, while the longer training duration did lead to slightly improved performances, the model exhibited clear signs of overfitting. The extended training duration also imposed a substantial computational cost without delivering proportional benefits in terms of generalization. Consequently, the decision was to exclude these alternative strategies in the final implementation.

3.2 Grid Search Training Phase

To identify the optimal architecture and hyperparameter values a grid search approach was implemented. This method involved testing a range of values for key components of the network, including the stride, the number of filters in the convolutional layers, as well as standard neural network hyperparameters such as learning rate and batch size.

To mitigate the risk of overfitting during the training phase, an early stopping callback was implemented, monitoring the validation loss. The validation loss provides a more reliable indication of the model's ability to generalize to unseen data, as it reflects the model's performance on a separate validation set not seen during training. The use of validation loss as a stopping criterion ensured that the model was optimized for generalization rather than for fitting the training data too closely.

It is important to note that the model was trained on both the original dataset and the dataset that was merged with synthetic data. The objective of this training was to ascertain whether the performance could be increased. Therefore the grid search was conducted on both using the same parameters grid, which included the following values: number of units in the convolutional layer set to 20 and 30, the stride of the kernel 1 and 2, the learning rate was 0.0001 and 0.00001 and batch size ranging from 32 to 64. The results of the phase revealed that the optimal network architecture varied depending on whether the model was trained on the original dataset or the augmented dataset, while the other hyperparameter values remained consistent across both cases. Specifically, the model trained on the augmented dataset achieved its best performance with a dense layer configuration of 30 filters per layer. In contrast, the model trained on the original dataset performed optimally with 20 filters per convolutional layer.

The observed differences in optimal network architecture may be attributed to the distinct representational demands imposed by the two datasets. Data augmentation, a process that introduces variability and diversity into the training data, likely increased the complexity of the feature space that the model was required to capture. Consequently, the model trained on the augmented dataset may have required a higher capacity, achieved here through 30 filters, to effectively learn and generalize the expanded patterns in the synthetic data. Despite the differences in network architecture, the optimal values for the remaining hyperparameters were identical for both models. These shared hyperparameters included a batch size of 64, a learning rate of 0.0001 and a stride of 1. As evidenced in the following table, the metrics derived from the optimal hyperparameter search on the validation set demonstrated a high degree of similarity. However, this similarity diminished when the metrics were tested on the test set.

	Validation set	
Data Augmentation	**Weighted F1 Score**	**Accuracy**
False	0.65	0.67
True	0.67	0.68

Table 1: Validation Phase Results

3.3 Testing Step

The testing phase was conducted on the independent test set using the optimal configurations identified during the grid search process. This phase served as a final evaluation to assess the generalization capabilities of the models on unseen data, ensuring that the selected configurations were not overfitted to the training and validation datasets. Notably, the model trained on the original dataset demonstrated superior performance on the test set, achieving a custom weighted f1 score of 0.74 compared to 0.61 for the model trained on the augmented dataset.

A detailed examination of the classification reports further elucidates the performance differences between the two models. For the augmented dataset, the model achieved a weighted average precision of 0.72, a recall of 0.48 and a traditional weighted f1 score of 0.53. These metrics reflect a model that struggles with precision and recall trade-offs, particularly in imbalanced classes. In fact, the smaller classes such as class 4 showed extremely poor recall (0.18) and standard f1 scores (0.05), indicating that the model was unable to generalize effectively for underrepresented classes. This trend suggests that the variability introduced by augmentation may have amplified the difficulty of capturing representative patterns for less frequent classes. On the other hand, the model trained on the original dataset achieved more balanced performance across the metrics, with a weighted average precision of 0.72, a recall of 0.63, and a standard weighted f1 score of 0.66. Furthermore, smaller classes, such as class 0 and class 1, demonstrated improved f1 scores (0.33 and 0.43, respectively), albeit still limited. This suggests that the model trained on the original dataset was better able to leverage the inherent structure of the data without the potential confounding effects of augmentation.

A key observation is the trade-off between recall and precision in both models, with the augmented dataset model showing a particular sensitivity to imbalances. While augmentation typically enhances generalization by exposing the model to a wider range of data variations, in this case, it may have inadvertently introduced noise or distorted the data distribution, complicating the learning process.

Class	Precision	Recall	F1-Score	Support
0	0.25	0.49	0.33	133
1	0.35	0.55	0.43	275
2	0.51	0.65	0.58	406
3	0.86	0.66	0.75	1896
4	0.06	0.13	0.08	39
Accuracy			0.63	2749
Macro Avg	0.41	0.50	0.43	2749
Weighted Avg	0.72	0.63	0.66	2749

Table 2: First Model Classification Report, Original Dataset

A thorough examination of the confusion matrices reveals that the models demonstrate varying degrees of success across the distinct classes. Class 3 emerges as the predominant class, exhibiting a considerable number of accurately predicted instances, though there are notable misclassifications. The remaining classes exhibit a reduced number of correct predictions, accompanied by significant confusion among neighboring classes. For instance, the true class 1 frequently misclassified into classes 3 and 0, suggesting potential feature overlap. Similarly, class 2 exhibits misclassifications distributed among classes 0, 3 and 4. For the augmented dataset, the comparison with the original dataset reveals that the overall accuracy and class distribution of predictions have shifted. In the original, class 3 was significantly dominant, while in the augmented, the correct predictions for this class were reduced. The redistribution indicates that modifications to the model are effective in addressing the bias toward class 3. Furthermore, class 1's correct predictions have increased, indicating improved learning for this class. Overall, while some improvement in class distribution and performance is evident, challenges in handling class imbalances and overlapping feature spaces persist.

Based on this analysis, it is evident how both models are highly influenced by imbalanced class distributions, which led to a strong bias toward predicting the majority label (class 3). Additionally, overlapping feature spaces among certain classes, particularly classes 0, 1 and 2, resulted in high misclassification rates. Insufficient representation of class 4 likely exacerbates its poor predictive performance.

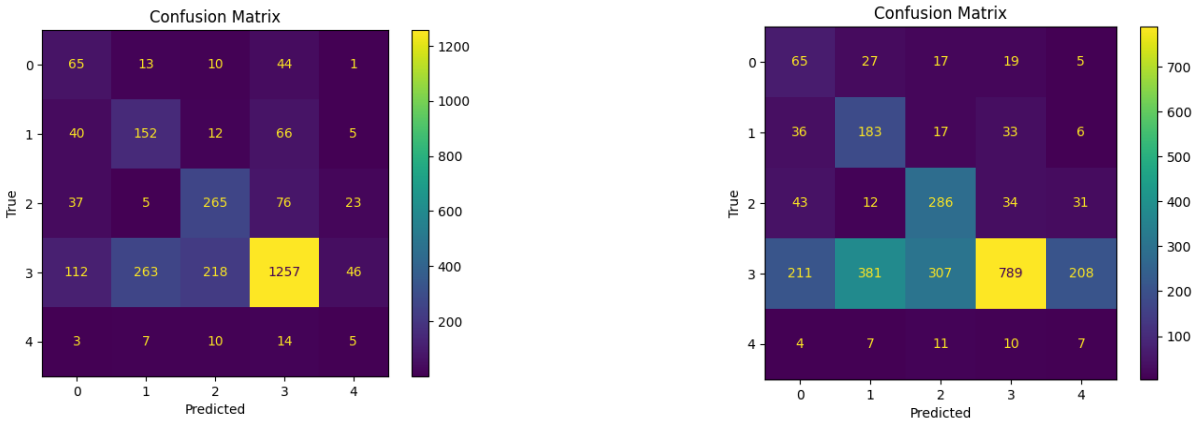

Figure 5: First Model Confusion Matrix: Original & Augmented Data

4 Second Model

4.1 Model Description

The second model developed is also based on a Convolutional Neural Network (CNN). To thoroughly assess the impact of varying architectures, this model integrates a range of components and parameters that differ from the previous model. These include an increasing number of filters in the convolutional layers and a densely connected classification component.

The model starts with an input layer tailored to accept images of a specific shape.
The initial stage consists of a series of convolutional layers, each equipped with a defined number of filters and a specific kernel size. To further enhance feature extraction, the number of filters is progressively doubled with each layer, allowing the model to learn a richer set of features as the depth of the network grows. Following each convolutional operation, a MaxPooling layer is applied. These layers serve to reduce the spatial dimensions of the feature maps, preserving the most prominent features while minimizing computational requirements. This down-sampling process ensures that the model focuses on the most salient aspects of the data, improving generalization. The selection of MaxPooling as opposed to AveragePooling, as the preceding model, was made with the objective of investigating whether this modification would result in enhanced performance variability. At the end of the convolutional stage, a global average pooling layer is incorporated. This component aggregates information across the spatial dimensions by computing the average value of each feature channel.

The classification part consists of one or more fully connected layers, depending on the passed parameter. These dense layers process the feature vector to make predictions. To address the issue of overfitting, dropout is applied after each dense layer, where a fraction, determined by the passed parameter, of neurons are randomly deactivated during training to enhance generalization. The final layer in the model is fully connected and uses a softmax activation function. In consideration of the preceding model, it is noteworthy that the loss function employed remains the categorical cross entropy. The optimizer continues to be Adam and an early stopping callback is consistently employed.

4.2 Grid Search Training Phase

The kernel size, the depth of the dense layers, the dropout rate and the number of units in the dense layers are not explicitly defined in the previous paragraphs because these were hyperparameters that influenced the model's performance. To determine the optimal configuration of these parameters, a grid search process was conducted. The hyperparameters tested in this study encompassed both convolutional and dense layer configurations to ensure a comprehensive exploration of the model's architecture. In the case of the convolutional layers, kernel sizes that exceeded the conventional size of 3 were examined, specifically 5 and 7. The decision to explore larger kernel sizes was motivated by the purpose of trying to capture broader spatial features in the input data, which could potentially enhance the model's ability to learn patterns. For the dense layers, several parameters were systematically varied. The initial number of neurons in each dense layer was set to either 32 or 64, with subsequent layers doubling the number of neurons in the previous layer. Additional hyperparameters tested included the dropout rate (0.25 and 0.5) and the total number of dense layers (1 and 2) in the classification component of the network.

A comprehensive grid search was conducted on both the original and augmented datasets. In contrast to the previous scenario, the configurations resulting from this grid search exhibited marked differences depending on the dataset used for training. Specifically, after evaluating 16 combinations of hyperparameters, the optimal configuration for the model trained on the original dataset achieved a custom weighted f1 score of 0.67 and an accuracy of 0.69 on the validation set. This configuration was characterized by the following parameters: dropout rate equal to 0.5, kernel size set to 7, number of dense layer units of 32 and only 1 dense layer before the last one with the soft-max. In comparison, the best performing configuration on the augmented dataset, which yielded a weighted f1 score of 0.69 and an accuracy of 0.70, was notably distinct. It comprised the following parameters: the dropout rate was limited to 0.25, the kernel size was designated as 5, the number of dense layer units was set to 64 and the number of dense layers before the last one was configured to 2.

The reasoning conducted on the previous model is applicable in this case as well. The observed discrepancy in optimal configurations across the datasets can be attributed to the inherent differences in their feature distributions and complexity. The augmented dataset, which has been enriched with additional variability through transformations, likely necessitates a more expressive model architecture to effectively capture the expanded feature space. The augmented dataset's optimal configuration, with its increased number of dense units (64 vs. 32) and dense layers (2 vs. 1), reflects these needs. The difference in dropout rates, 0.25 for the augmented dataset and 0.5 for the original dataset, suggests that the augmented dataset benefits from reduced regularization, possibly because the inherent diversity in the training samples mitigates overfitting.

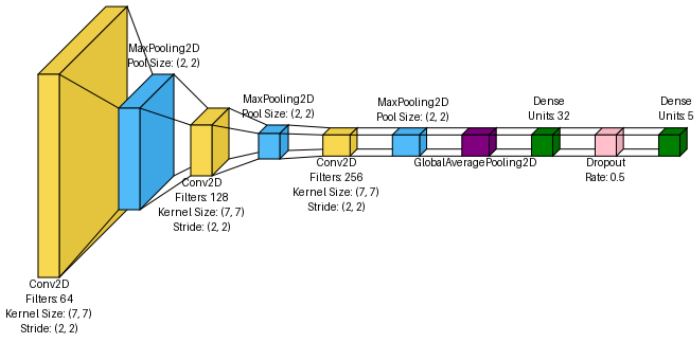

Figure 6: Optimal configuration for the Second CNN Model

4.3 Testing Step

The evaluation of the models was carried out on an independent test set using the hyperparameter configurations optimized during the grid search process. The results indicated that the model trained on the original dataset exhibited better generalization capabilities, achieving a custom weighted f1 score of 0.70, in contrast to the 0.64 obtained by the model trained on the augmented dataset.

An in-depth analysis of the classification reports highlights the variations in performance across the two models. The model trained without data augmentation achieved an overall accuracy of 0.61, with a standard weighted f1 score of 0.64. In contrast, the model trained with data augmentation obtained a lower accuracy of 0.54 and a classical weighted f1 score of 0.58.
Examining the class-specific metrics provides further insights into these results.
For the most prevalent label, designated as the third, both models demonstrated a high precision, with values of 0.88 being recorded. However, the recall for this class was significantly higher for the model trained without data augmentation (0.61 compared to 0.50), which led to a higher classic f1 score of 0.72 versus 0.64. This indicates that the model trained without augmentation was better at correctly identifying instances of the majority class while maintaining a balanced trade-off between precision and recall. For smaller classes, such as 0 and 4, the performance was generally poor for both models, however, the augmented dataset model exhibited marginally higher recall for class 4, suggesting that data augmentation might have improved the model's sensitivity to some extent for this underrepresented class. As was the case in the preceding model, the model that was trained without data augmentation appears to have taken advantage of the natural structure of the original dataset. This has led to more consistent generalization across classes, suggesting that the model trained on the original dataset was better equipped to capture the underlying structure of the data, translating to superior performance on unseen examples.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.22	0.53	0.31	133
1	0.35	0.61	0.44	275
2	0.47	0.69	0.56	406
3	0.88	0.61	0.72	1896
4	0.09	0.08	0.08	39
Accuracy			0.61	2749
Macro Avg	0.40	0.50	0.42	2749
Weighted Avg	0.72	0.61	0.64	2749

Table 3: Second Model Classification Report, Original Dataset

The analysis of the confusion matrices yields noteworthy insights, particularly due to the consistency observed between the matrices of the models trained with and without data augmentation. Despite the introduction of augmented data, the confusion matrices remain highly comparable to one another and also exhibit similarities to those obtained from the previous model.

As for the previous model the label 3 demonstrates the strongest performance, while class 2 shows moderate performance and the remaining exhibit relatively lower score. A notable observation is the substantial misclassification pattern between targets 2 and 3 that is also evident in this model. This finding confirms the presence of potential similarities in characteristics between these classes that the model is unable to differentiate.

A comparison of the matrices from the original dataset and the augmented dataset reveals minor discrepancies. The matrix derived from the original dataset demonstrates enhanced performance for class 3, with 1152 accurate predictions in comparison to the previous 948. Class 4 remains a challenging case in both scenarios. Notably, the misclassification between Classes 0 and 3 has been significantly reduced in the matrix from the original dataset, suggesting an improvement in differentiating between these classes. Given these observations, the results of the analysis led to the same conclusion seen for the previous model. So it confirmed several hypotheses: there must be underlying feature similarities between classes or noisy labels. The relatively poor performance on label 4 might indicate a need for targeted augmentation strategies or additional data collection for this specific class.

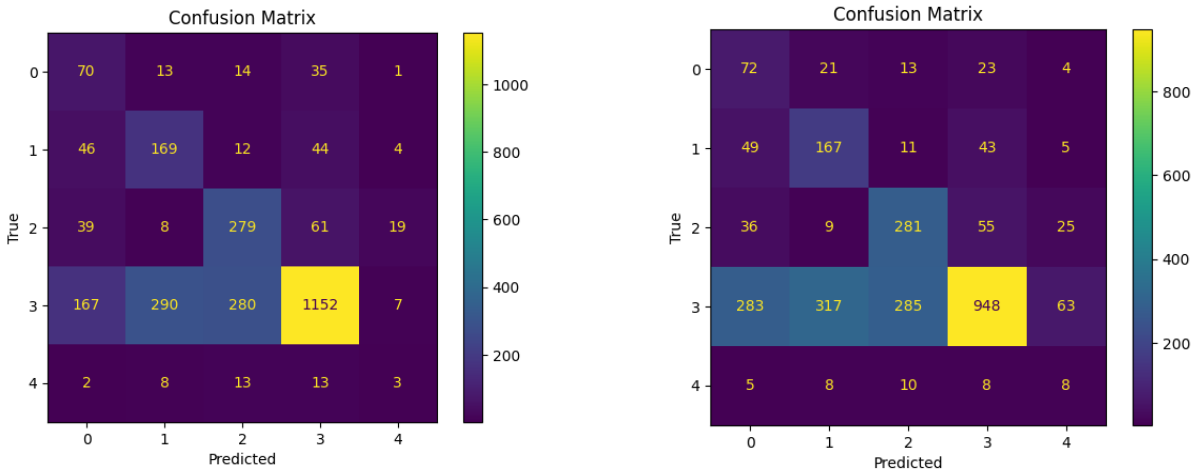

Figure 7: Second Model Confusion Matrix: Original & Augmented Data

5 Conclusions & Future Works

This report meticulously details the comprehensive analysis, implementation and evaluation of two convolutional neural network models designed to classify vehicular actions based on input images from a simulated driving environment. Despite the inherent challenges posed by an imbalanced and noisy dataset, the study has successfully demonstrated the efficacy of hyperparameter tuning and custom evaluation metrics in achieving reasonable performance levels.

The analysis revealed that both the models trained on the original dataset exhibited optimal performance. Specifically, the first model achieved the highest metrics values on the test set custom weighted f1 score of 0.74, compared to 0.70 of the second model and an accuracy of 0.63 compared to 0.60.

The comparison of models trained with and without data augmentation revealed nuanced trade-offs. While augmentation enhanced sensitivity to certain underrepresented classes, it achieved low generalization capabilities on the majority class and the test set. Conversely, models trained on the original dataset exhibited more consistent performance, leveraging the natural data structure to achieve superior results in both accuracy and weighted metrics. A key observation from these results is that while data augmentation is intended to enhance the model's generalization by introducing variability into the training data, it appears to have disrupted the balance of learning in this case. The augmented dataset may have introduced noise or led to over-representation of certain patterns, which could explain the decline in performance for the majority class and the overall reduction in weighted metrics. Another contributing factor might be the class imbalance, which could have been exacerbated by the augmentation process, leading to an overemphasis on certain classes at the expense of others.

The model's optimal performance was confirmed by loading it in the environment where the dataset images were captured. The findings were remarkable and unanticipated. Although metrics identified within the test set were satisfactory, the superior performance observed in the test conducted in the environment was not predicted.

Future work can explore several promising directions to enhance the robustness and generalization capabilities of the models. One potential avenue is the refinement of data augmentation strategies. Augmentation techniques should be more effectively aligned with the dataset's feature space. Class-specific augmentations, tailored to underrepresented classes while maintaining realistic data distributions, could mitigate the adverse effects observed in this study. Furthermore, the incorporation of advanced methods such as generative adversarial networks (GANs) to synthesize additional data may prove effective in addressing the class imbalance. Another critical area for improvement is feature engineering. Future research should prioritize the development of features that enable better differentiation between classes with overlapping feature spaces. Leveraging feature extraction techniques using pre-trained networks could facilitate the identification of more discriminative features, thereby enhancing the model's classification accuracy. Additionally, expanding the dataset to include more diverse samples, particularly for underrepresented classes, would significantly address the class imbalance issue and contribute to improved model generalization. An enriched and more diverse dataset would not only support robust training but also enhance the reliability of performance evaluation.