Lightweight YOLO for distracted driver detection on edge devices

Frank Zandamela[1*], *Dumisani* Kunene[1], *Vusi* Skosana[1], and *Gene* Stoltz[1]

[1]Council for scientific and Industrial Research, Optronic Sensor Systems, South Africa

Abstract. Edge AI, with its ability to process data locally on devices within vehicles, presents a promising approach to real-time driver monitoring. However, despite advancements in robust deep learning-based distracted driver detection, there is a critical gap in research on deploying these methods on edge devices. Real-world applications demand a balance between accuracy and real-time inference speed on resource-constrained devices. This work addresses this challenge by investigating the performance of a lightweight, human activity recognition-based distracted driver detection method. A comparative analysis study is conducted to compare the performance of four lightweight YOLO models. The study also explores the generalisability of the approach for driver distraction detection across four public datasets. Experimental results reveal that the tiny version of the YOLOv7 object detector provides the best balance between accuracy and inference speed. The algorithm achieved an average F1-score of 0.45 across four datasets and an average inference speed of 21.97 ms or 46 frames per second.

1 Introduction

Distracted driving poses a significant threat to road safety, prompting extensive research into detection methods. This is due to the consensus that if driver distractions can be detected and treated quickly through warnings, the incidence of accidents can be reduced. Many existing approaches use convolutional neural networks to detect distracting drivers, as deep learning is successful in solving real-world visual problems. Initially, deep learning distracted driver detection (DDD) methods focused on CNN feature extraction classification and then use fully connected layers [1], [2] or a multilayer perceptron (MLP) to classify the driver behaviour. However, CNN feature extraction methods often overfit the training data and subsequently fail to generalise well on new data not used for training [3]. As a result, the direction of research has switched towards developing robust distracted driver detection methods. Proposed robust methods include the use of the ensemble approach [4], [5], advanced data augmentation [6], [7], combining CNN features with HOG features [8], hybrid CNN-RNN approaches [9], [10], 3-dimensional CNNs [11], combining CNN features with human key points obtained using pose estimation algorithms [12], driver segmentation [13], [14], and driver ROI detection [15], [16].

---

[*] Corresponding author: fzandamela@csir.co.za

A recent study presented an approach that aims to enhance distracted driver detection by combining deep learning for DDD with human body activity recognition (HBAR) [17]. The dual approach allowed the HBAR method to capture more nuanced aspects of driver behaviour, leading to enhanced detection performance, boasting a state-of-the-art cross-dataset F1-score of 51% across three image datasets that were not used for training. Other researchers have also recently proposed a score-SoftMax classifier and a dynamic Gaussian smoothing supervision technique after they argued that the de facto One-Hot encoding results in CNNs being overconfident and subsequently learning background noise during training [18].

Despite advancements in robust DDD using deep learning, there is a critical gap in research on deploying these methods on edge devices. Real-world applications demand a balance between accuracy and real-time inference speed on resource-constrained devices. For effective DDD, the system must process data and generate accurate alerts instantly to enable swift corrective actions. Furthermore, achieving this balance requires consideration of factors like computational resources, power consumption, and cost. Research focused on developing lightweight DDD do exist, for example, Baheti *et al.* [19] proposed a modified version of the VGG architecture that uses depthwise separable convolutions instead of conventional convolutions. While Arefin *et al.* [8] included an additional 3×3 convolutional layer followed by a 2×2 max pooling layer to reduce the parameters of the AlexNet architecture. Recently, Liu *et al.* [20] proposed a lightweight DDD approach with distillation-based neural network architecture search and knowledge transfer framework. However, the detection accuracy and inference speed of the existing lightweight DDD approaches on AI edge devices remains unknown.

This study addresses this gap by evaluating the performance of a lightweight, HBAR-based distracted driver detection method deployed on a Jetson Xavier NX edge device. Specifically, we aim to answer two key research questions: 1) Which current state-of-the-art lightweight YOLO architecture is best suited for integration with the HBAR approach for driver distraction detection on the Jetson Xavier NX? 2) To what extent does the generalizability of the HBAR method suffer when implemented with a lightweight YOLO architecture?

The main contributions of this study are as follows:

- **Implementation of a lightweight YOLO-HBAR algorithm for a Jetson Xavier NX edge device**. We implement a lightweight DDD-HBAR algorithm using current a state-of-the-art YOLO model that is optimised for a Jetson Xavier NX edge device. Although current research is making strides towards more robust distracted driver detection, few to none of the existing studies focus on the implementing and evaluating the performance of DDD on an edge device. Insights drawn from this experiment will indicate best suitable lightweight YOLO model for the HBAR-based approach. These experiments will also demonstrate the model's adaptability and potential for real-world use cases.
- **Evaluating the cross-dataset performance of the YOLO-HBAR for edge DDD**. We present experimental results that focus on the cross-dataset performance on four public DDD datasets. The experiments provide valuable empirical data on the effectiveness of the lightweight YOLO-HBAR algorithm for DDD on AI edge devices. This data can be used for benchmarking and comparison with future advancements in lightweight object detection models.

2 Related work

2.1 Object detection

Object detection is a computer vision task that involves identifying and locating multiple objects within an image. There are two types of object detection algorithms based on deep learning: a two-stage and a single-stage object detector. The two-stage detector generates candidate regions that contain objects and then extracts features or classifies those regions using the CNN architecture. In contrast, single-stage detectors predict both the bounding boxes and class labels in a single pass through the CNN, bypassing the need for region proposals. The R-CNN family [21] is the most recognized group of two-stage detectors, while the YOLO family [22], [23], [24], [25], [26], [27] if the most well-known among single-stage detectors. This study focuses on the current state-of-the-art YOLO detectors due to their superior accuracy and inference speed compared to other detectors like Faster R-CNN [21] or EfficientDet [28].

In 2016, Redmon *et al.* [27] presented the first version of the YOLO family in their pioneering paper "You Only Look Once: Unified, Real-Time Object Detection" (YOLOv1). The core idea behind YOLO is to treat object detection as a regression task, enabling it to predict object bounding boxes and class probabilities directly from image pixels in a single forward pass through the network. Since the first YOLO model was introduced, researchers have published later versions. Most iterations aim to improve speed and accuracy by introducing new losses, different core CNN architectures, new concepts (anchor boxes, dense anchor boxes and pyramidal feature networks) and different training routines. For a detailed review of the YOLO family's development, readers are directed to [29].

2.2 Distracted driver detection on edge AI

In real-time driver monitoring, edge AI, with the ability to process data locally, presents a promising option. Few to none of the existing methods focus on developing lightweight DDD approach that can achieve a good balance between accuracy and inference speed. Chen *et al.* [30] used an ensemble approach that focused on fine-tuning three lightweight CNN architectures (Xception, MobileNet, and DenseNet) for DDD. Even though the authors reported a significant reduction in the number of parameters and an improved accuracy performance, the computational complexity and the cost of training multiple CNN architectures remains a concern. In addition, the authors did not evaluate the performance on an edge device nor report the processing speed of the method. Recently, Lou *et al.* [31] proposed a YOLOv5-based lightweight DDD approach that focuses on detecting common driver distractions such as using cell phone, drinking, and smoking. The authors modified the original YOLOv5s architecture by substituting the traditional convolutional layer with a Ghost convolution layer. However, the authors only focused on detecting driver distractions and performance of the proposed approach on an edge AI remains unknown. The trend in other research areas such as detecting small objects in ariel images, is also replacing the conventional convolutional layer with alternative convolutions such as the Ghost convolution [32], [33] and the pyramidal convolution (PyConv) [34].

3 Materials and methods

3.1 Distracted driver detection with human body activity recognition

The HBAR approach consists of two main steps as illustrated in Fig. 1. In the first step, the driver's body parts are detected, and the state of each body part is classified into a specific activity. The second step uses the detected activities to make the final prediction using a simple decision tree approach. The second phase of the driver's distraction classification problem is treated as a binary classification problem and focuses on determining whether the driver is distracted.

In the first step, the driver's head and hands are detected. The state of the head is classified as either "eyes on the road" or "eyes off the road." The position of the left and right hands in relation to the wheel is used to classify whether both hands are on the wheel, or one hand is on the wheel. Additionally, common distractions like using a cell phone, drinking, placing hands on the face, or talking on the phone are identified by detecting objects like a phone or a drinking bottle and classifying the activity based on the object. In the second stage of the approach, the final prediction was made by assessing two conditions: the road's eyes and both hands on the steering wheel. If both conditions are met, the driver is in a safe driving position. On the contrary, if one of the conditions is not met, the driver will be considered to be distracted.

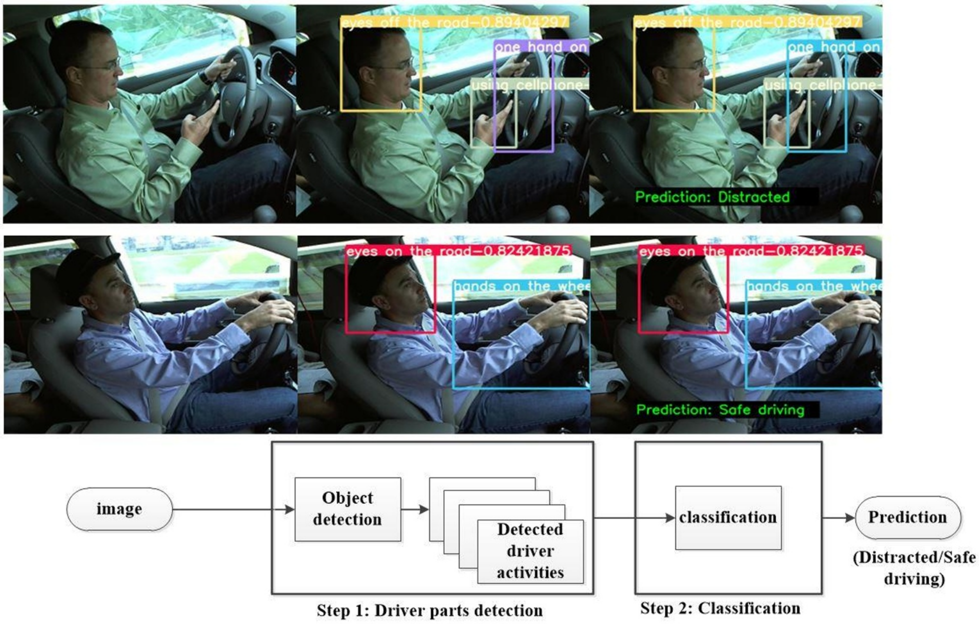
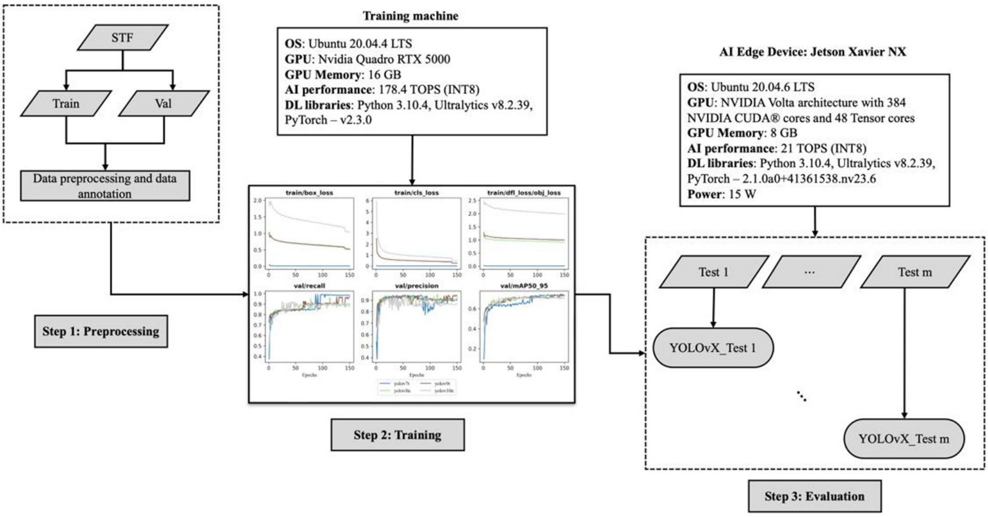


Fig. 1. Overview of the HBAR approach. **Step 1**: The driver's body parts are detected, and the activity of each body part is classified. **Step 2**: The detected activities are used to make a final prediction using a decision tree-based approach.

3.2 Implementation method

Fig. 2 shows a summary of the overall method that will be followed in this work. The STF dataset will be used training all the YOLO models selected. To allow for a fair comparison, the same modified STF dataset used to train the original DDD-HBAR approach [17] will be used. The modified STF dataset consists of 3346 randomly selected training images and 1000 images for validation. The original test split, composed of 2,247 images, was maintained for testing. The selected training and validation images were manually annotated with nine labels, including "eyes on the road", "eyes on the road", "hands on the wheel", "hands on the face", "no hands on the wheel", "one hand on the wheel", "using the mobile phone", "talking on the phone" and "drinking". The images were annotated using the labelimg[†] tool. It is worth noting that the size of the modified STF training dataset (3346 images) is small for a machine learning system that must be deployed in the real world. The decision to use only part of the STF training data set (3346/19173) is due to the time needed to annotate the images. It took a full month and three weeks to annotate the selected 4346 images (training and validation). However, work to train the DDD-BHAR approach on a large and diverse image dataset such as the 100-Driver dataset [35] is currently underway.



OS: Ubuntu 20.04.4 LTS
GPU: Nvidia Quadro RTX 5000
GPU Memory: 16 GB
AI performance: 178.4 TOPS (INT8)
DL libraries: Python 3.10.4, Ultralytics v8.2.39, PyTorch – v2.3.0

OS: Ubuntu 20.04.6 LTS
GPU: NVIDIA Volta architecture with 384 NVIDIA CUDA® cores and 48 Tensor cores
GPU Memory: 8 GB
AI performance: 21 TOPS (INT8)
DL libraries: Python 3.10.4, Ultralytics v8.2.39, PyTorch – 2.1.0a0+41361538.nv23.6
Power: 15 W

Fig. 2. Overall evaluation method.

In this study, four state-of-the-art YOLO models will be trained and evaluated to find a model that will lead to a DDD-HBAR balanced performance between speed and accuracy on a Jetson Xavier NX edge device. The selected YOLO models for analysis include YOLOv7-tiny [26], YOLOv8n[‡], YOLOv9t [25], and YOLOv10n [24]. Each model will only be trained on the modified STF dataset following the implementation process illustrated in Fig. 3. The models will be trained on a Linux computer equipped with an Nvidia Quadro RTX 5000 graphical processing unit (GPU) that has a 16GB memory and capable of 178.4 INT8 Tera Operations Per Second (TOPS).

To evaluate DDD accuracy and inference speed, each YOLO-HBAR approach will be tested on a Jetson Xavier NX edge device that has a GPU with an 8GB memory and capable of executing 21 TOPS. The setup of the Jetson device was completed using a Nvidia JetPack v5.1.3. Each algorithm will be evaluated on the five test datasets shown in Table 1. Four test sets (EZZ2021 [13], AUC2 [4], CSIR, 100-driver [35]) were included in addition to the STF test set to evaluate the cross-dataset performance of the algorithms. To ensure a consistent

---

[†] Labeling: https://github.com/HumanSignal/labelImg
[‡] No official research paper. Documentation can be found at https://github.com/ultralytics/ultralytics

evaluation across datasets with varying number of classes, a subset of 2210 images from the 100-driver dataset, containing classes shared with the other datasets (STF, EZZ2021, AUC2, and CSIR), was randomly selected to create a dedicated test set. Cross-dataset performance will indicate the ability of the algorithms generalise on new data. For detailed review of the datasets, the reader is referred to [36] and [35].

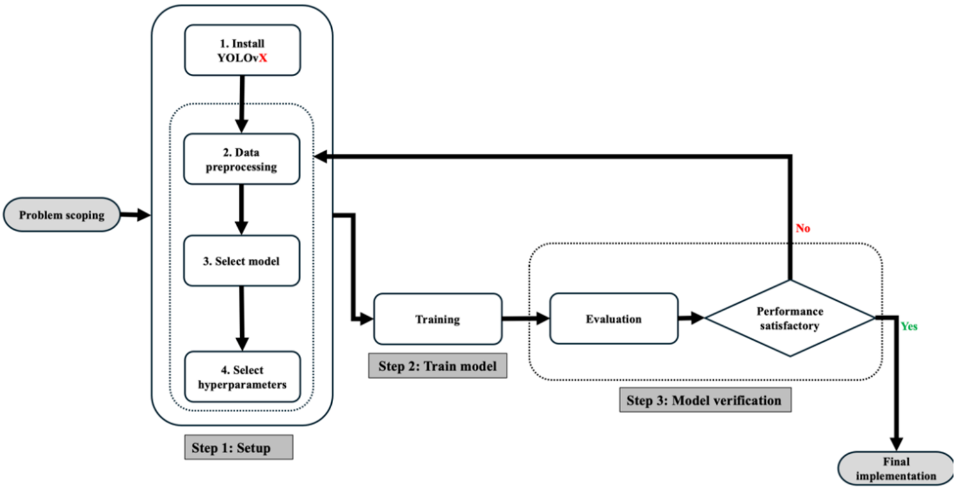


Fig. 3. Overall method used to implement the selected YOLO models.

Table 1. Characteristics of the distracted driver detection test datasets that will be used.

Dataset	Year	Environment	Type of distractions	Image samples
STF [37]	2016	Real	1 safe driving, 9 distracted activities	2247
EZZ2021 [13]	2021	Real	1 safe driving, 9 distracted activities	3716
AUC2 [4]	2019	Real	1 safe driving, 9 distracted activities	1074
CSIR[§]	2022	Real	1 safe driving, 9 distracted activities	512
100-driver [35	2023	Real	1 safe driving, 21 distracted activities	2210

3.3 Evaluation metrics

The cross-dataset performance of the selected algorithms is assessed using two quantitative metrics: accuracy and F1-score [38]. Accuracy, a commonly used and simple metric, measures the proportion of correct predictions made by the model out of the total observations in the test set. The F1-score, also known as the F-measure, is a crucial metric that represents the weighted harmonic mean of precision and recall, providing a balanced evaluation of a model's performance. The F1-score can be computed using E.q. (1).

[§] Dataset not yet published.

$$F1 - score = \frac{2(Precision \ \times Recall)}{Precision + Recall}, \tag{1}$$

In E.q. (1), recall measures the model's ability to identify all relevant cases and aims to reduce errors in naming positive cases as negative. In contrast, precision focuses on the accuracy of positive predictions and aims to reduce errors when classifying negative instances as positive. Precision and recall metrics can be calculated using

$$Precision = \frac{TP}{TP + FP}, \text{and} \tag{2}$$

$$Recall = \frac{TP}{TP + FN}. \tag{2}$$

The metrics TP, FP, and FN can be defined as follows [39]:

- **True positive (TP)**: Occurs when the model correctly predicts positive results, and the actual result is positive.
- **False positive (FP)**: Occurs when a model does not correctly predict positive results when the actual result is negative.
- **False negative (FN):** Occurs when a model incorrectly predicts a negative outcome when the actual outcome is positive.

The introduction of F1 score as a measure is based on its ability to provide a single comprehensive measure to evaluate the overall performance of classification models. The F1 score essentially achieves the balance between memory and precision and combines both aspects into a single value. The F1 score ranges from one to zero, with one score indicating complete accuracy and recall, and zero score indicating poor performance. Consequently, it provides a more detailed assessment of the model's performance.

In addition to the accuracy metrics, the inference speed of each model will also be measured in milliseconds (ms). For real-time inference the speed of an algorithm should at least be 33.33 ms or 30 frames per second (FPS) [40]. Further, the progression of loss curves, validation over prediction, recall curves or mean average precision (mAP) will be monitored to understand the level of training convergence that yields optimal results and avoid the overfitting and underfitting of the models. This observation will help determine how to limit training epochs per model and how the models react to hyperparameters during training and validation stages.

4 Experimental results and discussion

This section will provide experimental results that seek to address the following research questions:

- Which current state-of-the-art lightweight YOLO architecture is best suited for integration with the HBAR approach for driver distraction detection on the Jetson Xavier NX?
- To what extent does the generalizability of the HBAR method suffer when implemented with a lightweight YOLO architecture?

4.1 Training

The primary objective of this experiment is to train and evaluate the cross-dataset performance HBAR-based YOLO approach. All four models were trained on the annotated STF training dataset, based on respective lightweight YOLO models that were pretrained on the Microsoft COCO dataset. Each model was trained for 150 epochs using a 640 x 640 input image size, and a batch size of 8. The models were trained using the stochastic gradient descent (SGD) optimiser with an initial and final learning rate of 0.01 and 0.1, respectively. In addition, an intersection over union threshold value ($\mathrm{IoU_t}$) of 0.65 was used during training. These hyperparameters were obtained through a series of hyperparameter-tuning experiments. The other hyperparameters such as anchor threshold for YOLOv7-tiny (anchor_t) were set to their defaults.

Fig. 4 and Table 2 show the training performance of the four YOLO models. Based on the loss functions (box regression loss, classification loss, and object localisation loss) shown in Fig. 4, the YOLOv7t model has the lowest total loss compared to the other models. This suggests that the YOLOv7t model will have better detection accuracy. Table 2 summarises the validation performance of the models. All trained models achieved a mAP@50_95 that is above 0.50, suggesting that the models were able to learn features from the annotated STF dataset. In addition, the models were able to achieve precision and recall values that are above 0.85 with inference speeds that exceed 222 FPS on the training machine with an Nvidia Quadro RTX 5000 GPU. The inference speed of the models was calculated by adding the preprocessing time, inference time, and postprocessing (non-maximum suppression).

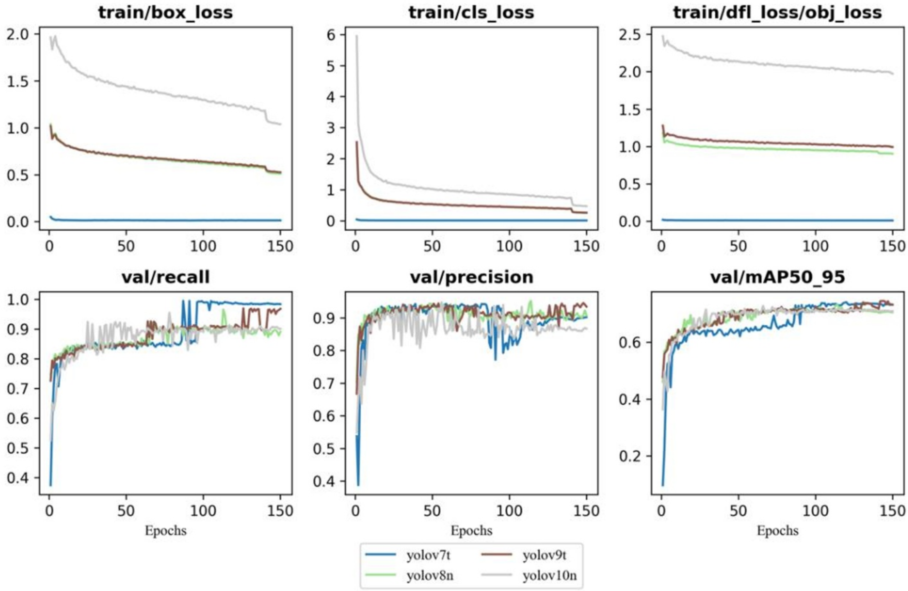


Fig. 4. Training and validation information of the YOLO models selected.

Table 2. Training performance metrics of the evaluated YOLO models. Speed measured using laptop with Quadro RTX 5000 and 16 GB per 640x640 image at a batch size of 8.

Model	**Parameters (M)**	**GFLOPS**	$\mathbf{mAP}^{\mathbf{val}}_{\mathbf{50\_95}}$	**Precision**	**Recall**	**Speed (ms)**
YOLOv7t	6.2	3.5	0.74	0.88	0.99	4.4
YOLOv8n	3	8.1	0.73	0.96	0.89	2.73
YOLOv9t	2	7.6	0.72	0.95	0.89	3.18
YOLOv10n	2.7	8.2	0.72	0.95	0.88	2.5

4.2 Baseline

The original HBAR algorithm trained on the modified STF training dataset using the YOLOv7 standard version will be used as a baseline for comparison purposes. The algorithm was trained for 150 epochs with an input image size of 640 x 640, an anchor threshold value of 9, and an IoU threshold value of 0.65. The other hyperparameters were set to their defaults. The training and validation performance metrics of the algorithm are summarised in Table 3. It is worth noting that these hyperparameters are not the same as the original implementation in [17].

Table 3. Training performance metrics of the standard YOLOv7. Speed measured using laptop with Quadro RTX 5000 and 16 GB per 640x640 image at a batch size of 8.

Model	**Parameters (M)**	**GFLOPS**	$\mathbf{mAP}^{\mathbf{val}}_{\mathbf{50\_95}}$	**Precision**	**Recall**	**Speed (ms)**
HBAR-YOLOv7	37.2	105.3	0.70	0.95	0.94	12.60

Table 4 shows a summary of the performance metrics (accuracy, F1-score, precision, recall, and processing speed) of the original implementation of the HBAR approach. The original HBAR-YOLOv7 algorithm was deployed on the Jetson Xavier NX edge device and tested across the five image datasets used in this work. Based on the results obtained, the original HBAR-YOLOv7 implementation obtained an average accuracy of 76.25%, F1-score of 0.40, and an average inference speed of 94.1 ms or 10 FPS.

Table 4. Performance metrics of the original HBAR-YOLOv7 algorithm when deployed on the Jetson Xavier NX edge device and inference run across the five test sets. * The ***Average*** a was calculated using only the EZZ2021, AUC2, CSIR and 100-driver test sets.

	Datasets					
Metric	STF	EZZ2021	AUC2	CSIR	100-driver	***Average****
Accuracy [%]	97	90	79	59	77	**76.25**
F1-score	0.85	-	0.44	0.22	0.23	**0.40**
Precision	0.82	0.0	0.61	0.71	0.63	**0.50**
Recall	0.88	0.0	0.35	0.13	0.14	**0.16**
Processing speed [ms]	110	90.1	90.1	90.2	10.1	**94.1**

4.3 Analysis and results

Table 5 shows the accuracy performance of the HBAR-YOLO algorithms across the five test sets. The average accuracy metric was calculated using only the EZZ2021, AUC2, CSIR, and 100-driver test sets because it is expected that the algorithms will perform well on the STF test set since it has the same distribution as the modified STF training dataset. Including the STF test set accuracy on the computation of the average value might inflate the final average metric. From the results, the following observations can be made:

- As expected, all algorithms perform well on the STF test set, with an average accuracy of 96.8%.
- The algorithms do not perform well on the CSIR custom and the 100-driver datasets, with an average of 56.88% or 69%, respectively.
- The HBAR-YOLOv8n algorithm achieved the best overall accuracy performance followed by the HBAR-YOLOv7t algorithm.
- Out of all the algorithms, the HBAR-YOLOv9t algorithm achieved the lowest overall cross-dataset accuracy, with an average of 69.25%.

Table 5. Accuracy scores of the HBAR-YOLO algorithms on the three datasets. * The ***Average*** accuracy was calculated using only the EZZ2021, AUC2, CSIR and 100-driver test sets.

			Accuracy			
Algorithm	STF	EZZ2021	AUC2	CSIR	100-driver	*Average**
HBAR-YOLOv7t	97	89	73	60	68	**72.5**
HBAR-YOLOv8n	96	90	80	57	70	74.25
HBAR-YOLOv9t	97	83	74	53	67	**69.25**
HBAR-YOLOv10n	97	85	75	57	71	**72.00**
Average	**96.8**	**86.8**	**75.5**	**56.8**	**69**	

For further analysis, the F1-score was used to compare the balanced performance of the algorithms. Table 6 shows the F1-scores of the algorithms. The precision and recall values that were used to calculate the F1-score are shown Table 7 and Table 8. Based on the F1-score results, the following observations can be made:

- As anticipated, the algorithms perform well on the STF test set, achieving an average F1-score of 0.86.
- Conversely, apart from the HBAR-YOLOv7t algorithm on the EZZ2021 test set, all approaches exhibit poor performance on the CSIR and EZZ2021 test sets, with an average of just 0.2.
- The HBAR-YOLOv7t algorithm achieved the best overall balanced performance, with an average F1-score of 0.45.
- The HBAR-YOLOv10n achieved the lowest average F1-score. A closer looker into the precision and recall values of the HBAR-YOLOv10n shows that the algorithm achieved precision and recall of zero on the EZZ2021 test set. This indicates the algorithm could not make a correct prediction for the positive class (safe driving) on the EZZ2021 test set.
- The algorithms achieve high precision, with an average score exceeding 0.50. However, recall scores are low, indicating a trade-off between accurately identifying driver distractions and correctly classifying safe driving behaviour. This suggests the algorithms might excel at catching distractions but miss instances of safe driving

Table 6. F1-scores of the HBAR-YOLO algorithms on the three datasets. * The ***Average*** F1-score was calculated using only the EZZ2021, AUC2, CSIR and 100-driver test sets.

			F1-scores			
Algorithm	STF	EZZ2021	AUC2	CSIR	100-driver	*Average**
HBAR-YOLOv7t	0.86	0.54	0.53	0.33	0.39	**0.45**
HBAR-YOLOv8n	0.85	0.04	0.30	0.15	0.33	**0.21**
HBAR-YOLOv9t	0.85	0.02	0.31	0.17	0.25	**0.19**
HBAR-YOLOv10n	0.86	-	0.08	0.13	0.23	**0.11**
Average	**0.86**	**0.2**	**0.31**	**0.2**	**0.30**	

Table 7. Precision scores of the HBAR-YOLO algorithms on the three datasets. * The ***Average*** precision was calculated using only the EZZ2021, AUC2, CSIR, and 100-driver test sets.

			Precision			
Algorithm	STF	EZZ2021	AUC2	CSIR	100-driver	***Average****
HBAR-YOLOv7t	0.81	0.43	0.45	0.65	0.37	**0.48**
HBAR-YOLOv8n	0.81	1.0	0.85	0.73	0.62	**0.80**
HBAR-YOLOv9t	0.81	1.0	0.60	0.71	0.57	**0.72**
HBAR-YOLOv10n	0.81	0.00	0.28	0.84	0.65	**0.44**
Average	**0.81**	**0.61**	**0.55**	**0.73**	**0.55**	

Table 8. Recall scores of the HBAR-YOLO algorithms on the three datasets. * The ***Average*** recall was calculated using only the EZZ2021, AUC2, CSIR, and 100-driver test sets.

			Recall			
Algorithm	STF	EZZ2021	AUC2	CSIR	100-driver	***Average****
HBAR-YOLOv7t	0.91	0.70	0.64	0.22	**0.41**	**0.49**
HBAR-YOLOv8n	0.90	0.02	0.18	0.08	**0.23**	**0.13**
HBAR-YOLOv9t	0.90	0.01	0.21	0.10	**0.16**	**0.12**
HBAR-YOLOv10n	0.91	0.00	0.05	0.07	**0.14**	**0.07**
Average	**0.91**	**0.18**	**0.27**	**0.12**	**0.24**	

Along with Accuracy and F1-score, processing speed is another crucial factor for real-world distracted driver detection systems. As shown in Table 9, the HBAR-YOLOv7t algorithm excels in processing speed, achieving an impressive 21.97 ms or 46 FPS. This indicates that it can handle real-time video processing efficiently. In contrast, the HBAR-YOLOv9t algorithm exhibits a slower processing speed of 43.55 ms or 23 FPS.

Table 9. Speed of the HBAR-YOLO algorithms on the three datasets.

			Processing speed [ms]			
Algorithm	STF	EZZ2021	AUC2	CSIR	100-driver	***Average***
HBAR-YOLOv7t	25.2	20.7	20.4	20.4	**23.17**	**21.97**
HBAR-YOLOv8n	34.3	37.2	37.1	37.0	**33.05**	**35.73**
HBAR-YOLOv9t	40.9	43.5	43.4	43.3	**46.66**	**43.55**
HBAR-YOLOv10n	32.5	35.2	35.2	35.1	**35.07**	**34.61**

Based the results and analysis presented above, the HBAR-YOLOv7t provides the good balance between detection accuracy (F1-score) and processing speed. Compared to the original implementation of the HBAR approach, there is an improvement of 12.5% on the

F1-score. In addition, the lightweight HBAR-YOLOv7t is 4.6x faster than the HBAR-YOLOv7t algorithm. However, more work is required to improve the poor recall score.

The 21.97 ms inference speed achieved by the algorithm can be attributed to the fact that YOLOv7t has less computational requirements (GFLOPS) compared to the other YOLO models as shown in Table 2. The HBAR-YOLOv10n algorithm has the worst F1-score and the HBAR-YOLOv9t algorithm is the slowest.

5 Conclusions and future work

This study presented work that seeks to address to research questions: 1) Which current state-of-the-art lightweight YOLO architecture is best suited for integration with the HBAR approach for driver distraction detection on the Jetson Xavier NX? 2) To what extent does the generalizability of the HBAR method suffer when implemented with a lightweight YOLO architecture? Based on the results obtained, the YOLOv7t architecture is the best suited for integration with the HBAR approach on the Jetson Xavier NX edge device. However, further work is required to improve the F1-score of the algorithm for real-world deployment.

The low recall obtained by the YOLOv7t-HBAR algorithm suggests that the performance of the feature extraction layers in the backbone must be improved. As a result, future work could focus on:

- Extracting latent information with Adaptive Training Sample Selection (ATSS). This will help models train on information that is not labelled on datasets
- Expand the receptive field of kernels on the backbone using Deformable Convolution to learn global information and the spatial relation of different body parts.
- Implement a balanced scaling of the CNN backbone as done with Efficient network architectures.

Future work will also focus on training the HBAR approach on a large and more diverse distracted driver detection dataset such as the 100-driver dataset.

References

1. V. B. Shankar and S. Thangam, *Distracted Driver Posture Recognition*, 2022 IEEE 3rd Global Conference for Advancement in Technology*,* GCAT 2022, 1–8, (2022)
2. C. Yan, F. Coenen, and B. Zhang, Driving posture recognition by convolutional neural networks, IET Computer Vision, **10**, 2, pp. 103–114 (2016)
3. F. Zandamela, T. Ratshidaho, F. Nicolls, and G. Stoltz, *Cross-dataset performance evaluation of deep learning distracted driver detection algorithms*, in Proceedings of 2022 Rapid Product Development Association of South Africa – Robotics and Mechatronics – Pattern Recognition Association of South Africa – South African Advanced Materials Initiative, RAPDASA-RobMech-PRASA-CoSAAMI, 9-11 November 2022, Somerset West, South Africa (2022)
4. H. M. Eraqi, Y. Abouelnaga, M. H. Saad, and M. N. Moustafa, Driver distraction identification with an ensemble of convolutional neural networks, J Adv Transp, **2019**, (2019)
5. C. Huang, X. Wang, J. Cao, S. Wang, Y. Zhang, A hybrid CNN framework for behavior detection of distracted drivers, IEEE Access, **8**, pp. 109335–109349 (2020)

6. C. Ou and F. Karray, Enhancing Driver Distraction Recognition Using Generative Adversarial Networks, IEEE Transactions on Intelligent Vehicles, **5**, 3, pp. 385–396 (2020)
7. J. Cronje and A. P. Engelbrecht, *Training convolutional neural networks with class based data augmentation for detecting distracted drivers*, in Proceedings of the 9th International Conference on Computer and Automation Engineering, ICCAE '17, pp. 126–130 (2017)
8. M. R. Arefin, F. Makhmudkhujaev, O. Chae, and J. Kim, *Aggregating CNN and HOG features for Real-Time Distracted Driver Detection*, in IEEE International Conference on Consumer Electronics, ICCE, pp. 12–14 (2019)
9. C. Streiffer, R. Raghavendra, T. Benson, and M. Srivatsa, *DarNet: A deep learning solution for distracted driving detection*, in Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference: Industrial track, 11-15 December 2017, Las Vegas, Nevada, United States (2017)
10. J. Mafeni Mase, P. Chapman, G. P. Figueredo, and M. Torres Torres, *A Hybrid Deep Learning Approach for Driver Distraction Detection*, in 2020 International Conference on Information and Communication Technology Convergence, ICTC, 21-23 October 2020, Jeju, South Korea (2020)
11. F. Nel and M. Ngxande, *Driver Activity Recognition through Deep Learning*, in Proceedings of 2021 Southern African Universities Power Engineering Conference/Robotics and Mechatronics/Pattern Recognition Association of South Africa, SAUPEC/RobMech/PRASA, 27-29 January 2021, Potchefstroom, South Africa (2021)
12. M. Cetinkaya and T. Acarman, *Driver activity recognition using deep learning and human pose estimation*, in 2021 International Conference on Innovations in Intelligent Systems and Applications, INISTA, 25-27 August 2021, Kocaeli, Turkey (2021)
13. A. Ezzouhri, Z. Charouh, M. Ghogho, and Z. Guennoun, Robust Deep Learning-Based Driver Distraction Detection and Classification, IEEE Access, **9**, pp. 168080–168092, (2021)
14. M. Leekha, M. Goswami, R. R. Shah, Y. Yin, and R. Zimmermann, *Are you paying attention? Detecting distracted driving in real-time*, in 2019 IEEE Fifth International Conference on Multimedia Big Data, BigMM, 11-13 September 2019, Singapore, (2019)
15. B. T. Dong and H. Y. Lin, *An On-board Monitoring System for Driving Fatigue and Distraction Detection*, in 2021 22nd IEEE International Conference on Industrial Technology, ICIT, 10-12 March 2021, Valencia, Spain (2021)
16. F. Sajid, A. R. Javed, A. Basharat, N. Kryvinska, A. Afzal, and M. Rizwan, An efficient deep learning framework for distracted driver detection, IEEE Access, **9**, pp. 169270–169280 (2021)
17. F. Zandamela, F. Nicolls, D. Kunene, and G. Stoltz, Enhancing distracted driver detection with human body activity recognition using deep learning, South African Journal of Industrial Engineering, **34**, 4 (2023)
18. C. Duan, Z. Liu, J. Xia, M. Zhang, J. Liao, and L. Cao, Enhancing Cross-Dataset Performance of Distracted Driving Detection with Score Softmax Classifier and Dynamic Gaussian Smoothing Supervision, IEEE Trans on Intelligent Vehicles, (2024)
19. B. Baheti, S. Talbar, and S. Gajre, Towards Computationally Efficient and Realtime Distracted Driver Detection with MobileVGG Network, IEEE Trans on Intelligent Vehicles, **5**, 4 (2020)

20. D. Liu, T. Yamasaki, Y. Wang, K. Mase, and J. Kato, Toward Extremely Lightweight Distracted Driver Recognition with Distillation-Based Neural Architecture Search and Knowledge Transfer, IEEE Trans on Intelligent Transportation Systems, **24**, 1 (2023)
21. S. Ren, K. He, R. Girshick, and J. Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, IEEE Trans Pattern Anal Mach Intell, **39**, 6 (2017)
22. J. Redmon and A. Farhadi, *YOLO9000: Better, faster, stronger*, in IEEE Conference on Computer Vision and Pattern Recognition*,* CVPR 2017, 21-26 July 2017 Honolulu, HI, United States (2017)
23. J. Redmon and A. Farhadi, YOLOv3: An incremental improvement, ArXiv, (2018)
24. A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, and G. Ding. Yolov10: Real-time end-to-end object detection, arXiv preprint arXiv*:2405.14458* (2024).
25. C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information, arXiv (2024)
26. C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 18-22 June 2023, Vancouver, Canada (2023)
27. J. Redmon, S. Divvala, R. Girshick, A. Farhadi, *You only look once: Unified, real-time object detection*, in Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 27-30 June, Las Vegas, Nevada (2016)
28. M. Tan, R. Pang, Q. V. Le, *EfficientDet: Scalable and efficient object detection*, in Proceedings of the IEEE Computer Vision and Pattern Recognition, CVPR, June 16-18, (2020)
29. P. Jiang, D. Ergu, F. Liu, Y. Cai, B. Ma, A Review of Yolo Algorithm Developments, Procedia Comput Sci, **199** (2021)
30. D. Chen, Z. Wang, J. Wang, L. Shi, M. Zhang, and Y. Zhou, Detection of distracted driving via edge artificial intelligence, Computers and Electrical Engineering, October, **111** (2023)
31. C. Lou, X. Nie, Research on Lightweight-Based Algorithm for Detecting Distracted Driving Behaviour, Electronics (Switzerland), **12**, 22 (2023)
32. J. Cao, W. Bao, H. Shang, M. Yuan, Q. Cheng, GCL-YOLO: A GhostConv-Based Lightweight YOLO Network for UAV Small Object Detection, Remote Sens (Basel), **15**, 20 (2023)
33. M. Hu, Z. Li, J. Yu, X. Wan, H. Tan, Z. Lin, Efficient-Lightweight YOLO: Improving Small Object Detection in YOLO for Aerial Images, Sensors, **23**, 14, July (2023)
34. I. C. Duta, L. Liu, F. Zhu, and L. Shao, Pyramidal Convolution: Rethinking Convolutional Neural Networks for Visual Recognition, Arxiv (2020)
35. J. Wang *et al.*, 100-Driver: A Large-Scale, Diverse Dataset for Distracted Driver Classification, IEEE Transactions on Intelligent Transportation Systems, **24**, 7 (2023)
36. F. Zandamela, “Enhancing Cross-Dataset Performance in Distracted Driver Detection using Body Part Activity Recognition,” Thesis, University of Cape Town, 2024. Accessed: Jun. 30, 2024. [Online]. Available: http://www.dip.ee.uct.ac.za/publications/theses/MScFrankZ.pdf
37. “State Farm Distracted Driver Detection | Kaggle.” Accessed: Mar. 28, 2022. [Online]. Available: https://www.kaggle.com/c/state-farm-distracted-driver-detection

38. Y. Sasaki, “The truth of the F-measure,” 2007. Accessed: Jun. 30, 2024. [Online]. Available: https://www. cs. odu. edu/mukka/cs795sum09dm/Lecturenotes/Day3/F-measure-YS-26Oct07. pdf
39. P. Padilla, S. L. Netto, and E. A. B. da Silva, A Survey on Performance Metrics for Object-Detection Algorithms, in 2020 International Conference on Systems, Signals and Image Processing, IWSSIP, 01-03 July 2020, Niteroi, Brazil (2020)
40. M. Capra, B. Bussolino, A. Marchisio, G. Masera, M. Martina, and M. Shafique, Hardware and Software Optimizations for Accelerating Deep Neural Networks: Survey of Current Trends, Challenges, and the Road Ahead, IEEE Access, **8** (2020)