Using Images from a Video Game to Improve the Detection of Truck Axles

Leandro Arab Marcomini, Andre Luiz Cunha

October 1, 2025

Abstract

Convolutional Neural Networks (CNNs) traditionally require large amounts of data to train models with good performance. However, data collection is an expensive process, both in time and resources. Generated synthetic images are a good alternative, with video games producing realistic 3D models. This paper aims to determine whether images extracted from a video game can be effectively used to train a CNN to detect real-life truck axles. Three different databases were created, with real-life and synthetic trucks, to provide training and testing examples for three different You Only Look Once (YOLO) architectures. Results were evaluated based on four metrics: recall, precision, F1-score, and mean Average Precision (mAP). To evaluate the statistical significance of the results, the Mann-Whitney U test was also applied to the resulting mAP of all models. Synthetic images from trucks extracted from a video game proved to be a reliable source of training data, contributing to the performance of all networks. The highest mAP score reached 99%. Results indicate that synthetic images can be used to train neural networks, providing a reliable, low-cost data source for extracting knowledge.

1 Introduction

Neural networks enable engineers to tackle intricate problems, analyze large datasets, and develop intelligent systems that can automate processes, enhance decision-making, and improve overall system performance. With their ability to handle non-linear relationships and adapt to changing conditions, neural networks contribute significantly to advancing engineering practices and fostering innovation in diverse applications. These computational models, inspired by the structure and functioning of the human brain, are capable of learning and making predictions from complex data. They are employed for tasks like pattern recognition and fault detection [1], optimization and control systems [2], and image processing [3].

Various transportation engineering applications use neural networks, ranging from energy storage systems to passenger demand forecasting and aircraft maintenance. In [4], the authors developed short-term passenger demand forecasting models for light rail services using multi-layer perceptron (MLP) models, with each time slot handled independently to eliminate significant seasonality. Artificial Neural Networks have also been used for automatic number plate recognition systems [5] and as an auxiliary factor in planning transportation routes [6]. In vision-based systems, Convolutional Neural Networks (CNNs) were used to propose a flexible neural network for fast and accurate road scene perception [7], which is crucial for autonomous vehicles. CNNs were also used to detect and count truck axles in images [8], an essential task in characterizing the truck fleet for planning and operating transportation systems.

CNNs, commonly used in image classification and object detection, require vast amounts of data to extract knowledge from [9]. An image dataset is critical for the training process of CNNs because it provides the network with patterns and features to learn from. Therefore, data collection is a crucial step in the training process of convolutional neural networks [10].

Data collection involves three main steps: acquiring, labeling, and improving existing data or models [11]. Data acquisition involves collecting raw data from various sources, such as cameras, sensors, or existing databases. Data labeling involves annotating the collected data with relevant labels or tags to provide the network with ground truth information. Improving existing data or models consists in refining the collected data or models to improve their quality and accuracy. Data collection is a challenging task that requires significant effort and resources [12].

To overcome the issue of limited data availability, there are different approaches for training deep networks. These include data augmentation, where the existing data is modified through various image transformations, or using pre-trained networks that have already been trained on similar tasks as a starting point for the original problem [13]. Another approach is to create artificial training samples using computer-generated (CG) images. This involves using computer models to generate synthetic data, which can be added to the training set, providing more diverse data for training the network [14]. An example of synthetic data compared to real-world data can be seen in Figure 1.

Figure 1: Example of synthetic and real-world data. The truck on the top is a rendered 3D model. The truck on the bottom is a real-life truck.

The main objective of this paper is to examine whether synthetic images from a video game can be effectively used to train a CNN for detecting real-life objects. To accomplish this, 27 different neural networks were trained using a combination of synthetic and real-world images of trucks, with a focus on detecting truck axles. All the networks were based on the YOLO model but were trained using different combinations of images and base weight structures. The results were evaluated using four commonly used metrics in neural networks: (1) precision, (2) recall, (3) mAP, which considers the accuracy of the object's location prediction, and (4) F1-score, a balanced metric of precision and recall.

2 Convolutional Neural Networks

Convolutional neural networks are a class of deep neural networks that are particularly well-suited for classifying images [15]. CNNs use an ad-hoc architecture inspired by our understanding of processing within the visual cortex, and they leverage spatial information to identify patterns in images. CNNs comprise one or more layers of two-dimensional filters with activation functions and, usually, down-sampling. Each layer learns to detect different features in the input data, and the network as a whole can learn to abstract relevant information automatically while the data is being processed [16]. In Figure 2, it is possible to see a common architecture of a CNN, with one convolution layer extracting features, followed by a fully connected multi-layer perceptron, known as the classification layer, to classify images based on given classes.

Several models have been developed to enhance the performance and efficiency of CNNs. One prominent model is the LeNet-5 [18], which was one of the earliest CNN architectures, comprised of multiple convolutional and pooling layers. This model paved the way for more advanced architectures

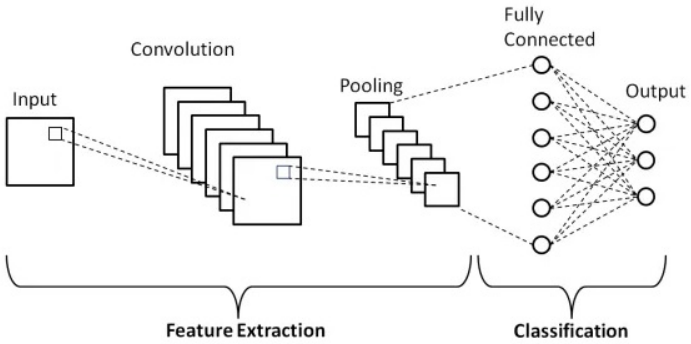


Figure 2: CNN architecture, with feature extraction happening on the convolution layers, and classification provided by a fully connected MLP. Source: [17].

such as AlexNet [19], which introduced the concept of deep CNNs by employing multiple layers with millions of parameters. Another notable model is VGGNet [20], which emphasizes the importance of using smaller filter sizes and deeper architectures. GoogLeNet [21], also known as Inception, introduced the concept of 'inception modules' that employed multiple filter sizes in parallel, effectively capturing different scales of features. The ResNet model [22] introduced residual connections to alleviate the vanishing gradient problem and enable the training of extremely deep networks. These models have significantly contributed to the development of CNN architectures, enhancing their capabilities in tasks such as image classification, object detection, and semantic segmentation.

An example of a modern algorithm that uses the ResNet algorithm is 'You Only Look Once' (YOLO). YOLO [23] is a real-time object detection system that uses a single neural network to predict bounding boxes and class probabilities for objects in an image. YOLO divides the image into a grid, and each grid cell predicts a fixed number of bounding boxes, along with the class probabilities for each box. YOLO has gone through many development iterations, resulting in several different versions.

YOLOv3 [24] uses a feature extractor based on a residual network and a spatial pyramid pooling (SPP) module to capture features at different scales. YOLOv3 also uses multi-scale training, where the network is trained on images of different sizes, to improve its ability to detect objects of different sizes.

The architecture of YOLOv3, which is the basis for all different versions, can be seen in Figure 3.

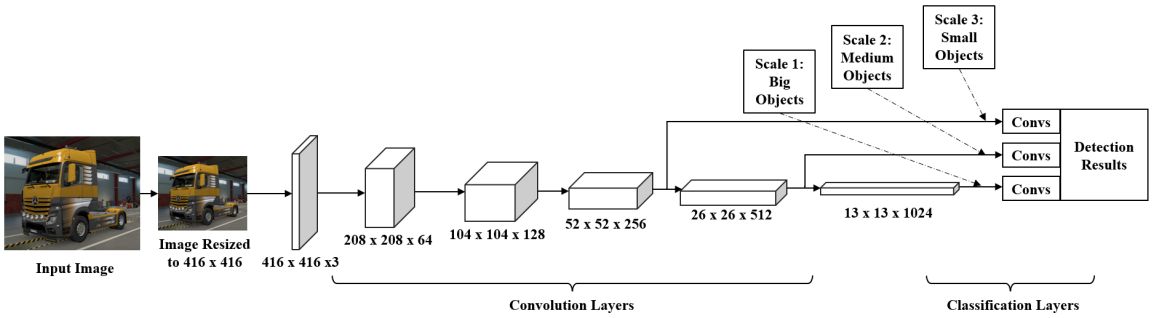


Figure 3: YOLOv3 layers and architecture, with several convolution layers and different classification scales to detect different-sized objects. Source: adapted from [25].

YOLOv8 [26], released in 2023, uses a CSP (Cross Stage Partial) Darknet backbone with bottleneck layers, reducing the overall computational complexity from previous models. This backbone architecture has a focus layer, which splits the input channels for early feature detection, and still uses a SPP module for multi-scale detection.

YOLOv11 [27] was released in 2024 and introduced several architectural enhancements to improve object detection performance. The backbone incorporates C3K2 blocks, which uses smaller 3×3 convolutional kernels to optimize feature extraction and computational efficiency. It still uses an SPP module to improve the detection of objects of different sizes. Additionally, the C2PSA (Cross Stage

Partial with Spatial Attention) block is integrated to improve spatial attention, enabling the model to focus on critical regions within an image [28].

Although all models follow the base structure for YOLO, the number of layers and parameters vary from model to model. A summary is presented on Table 1.

Table 1: YOLO Models: Layers and Parameters

Model	**Layers (Approx.)**	**Parameters (Approx.)**
YOLOv3-Tiny	23	8.7 million
YOLOv3	106	61.5 million
YOLOv3-SPP	110	63 million
YOLOv8n	168	3.2 million
YOLOv8l	268	43.7 million
YOLOv8x	328	68.2 million
YOLOv11n	150	2.8 million
YOLOv11l	250	42.5 million
YOLOv11x	310	66.8 million

Since CNNs require large amounts of labeled data to learn and recognize patterns in images accurately, insufficient examples can lead to overfitting, where the network becomes too specialized to the training set of images and performs poorly on new data [29]. Because of that, data collection is crucial for neural networks because it provides the necessary information for the network to learn and make accurate predictions.

3 Data Collection

Data collection is gathering and measuring information on variables of interest in an established systematic fashion to systematically answer stated research questions, test hypotheses, and evaluate outcomes [30]. It is a critical step in research and analysis, providing the foundation for decision-making and problem-solving. Data collection can be done through various methods, such as surveys, interviews, observations, and experiments, and it can be quantitative or qualitative.

In transport engineering, data collection is essential for (1) planning and design since it helps in understanding the current state of the transport system and identifying areas that need improvement. This information is then used to plan and design new transport infrastructure and services [31]; (2) monitoring and maintenance, where data collection is used to monitor the performance of the transport system and identify areas that require maintenance or repair. This monitoring ensures the safety and efficiency of the transport system [32]; and (3) research and development, where data collection is used to create databases to understand the behavior of the transport system and develop new technologies and solutions to improve it [33, 34].

One way to collect data in transport engineering is by using remote sensors, such as cameras. Images can be used to analyze and detect cracks in the pavement [35], to monitor urban traffic using UAVs [36], and to detect vehicles in traffic, to avoid collisions [37]. The authors commonly apply neural networks to process large amounts of data to conclude the researched subject. Figure 4 shows a schematic of the process. Data collection is the first step, and all subsequent steps depend on it.

Collecting real-world data can be time-consuming, expensive, and dangerous [39]. In transport engineering, collecting vehicle images generally involves cameras mounted on the side of high-speed highways, exposed to vehicle fumes, and on different weather and lighting conditions [40, 8]. That is why synthetic data can be a valuable choice to increase the amount of available data, be it because of cost or time restrictions or to improve the safety of all people involved.

3.1 Synthetic Data

Synthetic data refers to artificially generated data produced by computer programs that simulate real-world data [41]. It is becoming increasingly important to neural networks due to the limitations

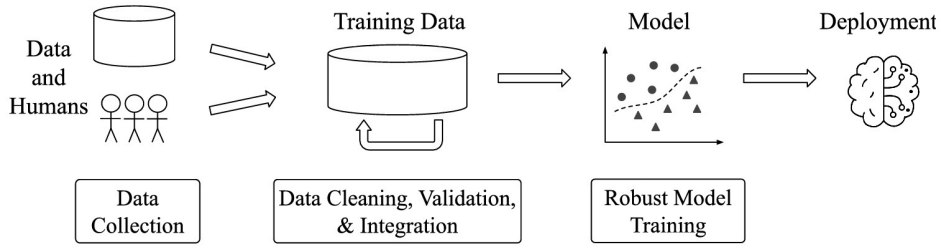


Figure 4: The process of using neural networks and the importance of data collecting. Source: adapted from [38].

of real-world data, such as limited quantity, quality, and diversity. Data collection is a significant bottleneck in machine learning, and synthetic data can help overcome this limitation.

Artificial data can be generated at a lower cost and faster rate than real-world data, making it more feasible for large-scale training of neural networks [42]. It can handle the variability in real-world data by randomizing the simulator's parameters, such as lighting, pose, and object textures, to force the neural network to learn the essential features of the object of interest. Augmenting real-world data by generating additional samples completing the data set [43] is also possible.

Synthetic data has been used in various applications of neural networks in several fields of knowledge, such as natural scene text recognition [44], three-dimensional nuclear segmentation of biological images [45], and automatic license plate recognition [46].

However, the effectiveness of synthetic data depends on how well it represents real-world data and how well it generalizes to new data [47]. Therefore, it is vital to design and evaluate synthetic data generation methods carefully and to combine them with real-world data in a balanced way to achieve optimal performance of neural networks.

3.2 Simulators

Simulators are essential in transport engineering, allowing engineers to test and evaluate new designs and technologies in a safe and controlled environment. They replicate real-world scenarios, enabling engineers to study and predict the behavior of transportation systems under different conditions. Simulators can be used to test new vehicle designs [48], to model driver's behavior [49], to develop traffic management systems [50], and can help engineers identify potential problems and optimize performance before deploying new systems in the real world.

Video games are increasingly being used as a source of data to train neural networks applied to transport engineering and are viewed as simulators. In [51], the authors use a city builder game to visualize urban modeling techniques. In [52], a video game is used to train and test AI drivers' capabilities, being able to supplement their human counterparts.

Euro Truck Simulator 2 [53], a video game that simulates the experience of driving a truck across Europe, can be a valuable resource for gathering truck images and data for the field of transportation engineering. The game's simulation of different driving conditions and scenarios could provide data on truck performance and driver behavior, informing the design of safer and more efficient transportation systems for heavy vehicles. Additionally, the high level of detail and realism in replicating various truck models, road networks, and driving conditions could provide a rich dataset for analysis. Figure 5 shows a realistic model of a truck extracted from the game.

To the best of our knowledge, there are no previous works using video games as a source of synthetic images to train a truck axle detection neural network. This gap presents a unique opportunity for research, as leveraging synthetic truck data from video games could provide a cost-effective and scalable solution for generating diverse training datasets, particularly in scenarios where real-world truck images are limited or difficult to obtain.

4 Experiment

Since YOLO is a stable algorithm developed for several years, its parameters are well-studied and understood. Because of that, it was chosen as a base to test the ability to use synthetic data extracted

Figure 5: 3D model of a truck from the video game Euro Truck Simulator 2.

from Euro Truck Simulator 2 to train a neural network. In order to compare results and fulfill the objective, the experiment was divided into four parts: (1) building a database of images, (2) training three variations of the YOLO algorithm, each with three different base models, (3) applying the models on a fixed number of images, (4) evaluating the results with five different metrics. Figure 6 shows the method applied to this paper.

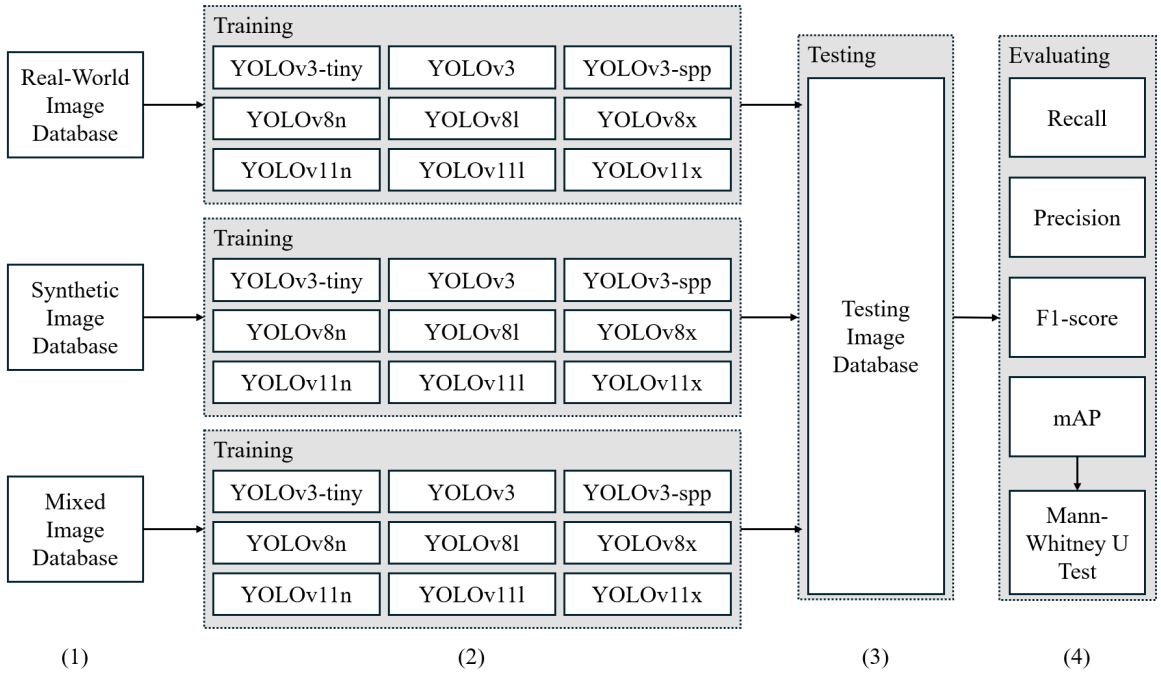


Figure 6: All four steps of the experiment, from the creation of the database, (1), to the evaluation of all metrics, (4).

4.1 Database

For this paper, four different databases were created. It is important to note that all databases used for training purposes must have a similar number of images and objects of interest to eliminate interference from any variables that should not influence the results. However, the database used during the testing

phase may contain a different number of images as long as it is kept the same for all models to compare results. Furthermore, the objects of interest in this paper are truck axles. Because of that, truck images where axles are visible were selected.

The first database contains 346 real-world images of trucks extracted from videos on two different highways in Brazil. The videos were recorded on separate days, including a variation of weather and lighting conditions, with the camera positioned on the side of the road, on the right-side lateral clearance, as seen in Figure 7.

Figure 7: Example of trucks from the first database containing real-life trucks.

The second database contains 326 synthetic images from the Euro Truck Simulator 2 video game, which are available on the community site for the game [54]. The images were extracted and selected to contain at least one axle in plain view so the neural network could extract the necessary information. Images are usually artistic since people mostly use the community site to showcase their trucks. Even though that is the case, truck models are depicted in realistic form, and truck axles are mostly visible. Examples of trucks used in the database can be seen in Figure 8.

The third database was constructed with mixed images extracted from both previous databases, with 175 images from real-life trucks and 175 from synthetic 3D trucks. The number of images was chosen to maintain a similar number of truck axles in all databases, with approximately the same number of images.

The fourth database, the testing set of images, was built using only images from the real world,

Figure 8: Example of trucks from the second database containing synthetic images with 3D models.

all different from those in the first database since the application is intended to detect truck axles in real-life truck images. It is composed of 36 images with 119 truck axles. A summary of all databases can be seen in Table 2.

Table 2: Training and testing databases, with the number of images and axles.

	First Database Real-life Trucks	**Second Database Synthetic Trucks**	**Third Database Mixed Trucks**	**Testing Database**
Images	346	326	350	36
Axles	1184	1148	1176	119

4.2 Training

It is common to use pre-trained weight files as a starting point for all neural networks to enhance the efficiency of the training process. For the YOLOv3, YOLOv8 and YOLOv11 architectures used in this paper, three specific pre-trained weight files were utilized for each version: YOLOv3-tiny, YOLOv3, and YOLOv3-spp, YOLOv8n, YOLOv8l and YOLOv8x, and YOLOv11n, YOLOv11l and YOLOv11x. Incorporating these pre-trained weights as a base can expedite the training procedure while benefiting from the knowledge encoded in these models.

YOLOv3-tiny is a lightweight variant optimized for faster inference on resource-constrained devices, with 23 layers and approximately 8.7 million parameters [55]. Compared with the standard YOLOv3, it has fewer layers and parameters, resulting in smaller model size and quicker inference speed [56].

However, YOLOv3-tiny sacrifices some accuracy in object detection, particularly for small objects or objects at longer distances.

YOLOv3 is the standard version of the applied YOLO algorithm. It is deeper and more complex than YOLOv3-tiny, with 106 layers and around 61 million parameters [24]. YOLOv3 performs well across different object sizes and can accurately detect a wide range of objects. However, due to its larger size, it is slower during inference.

YOLOv3-spp is an enhanced version of YOLOv3, with small changes in the number of parameters (63 million) and layers (110), incorporating Spatial Pyramid Pooling (SPP). This module allows the network to capture information at multiple scales by dividing the input image into grids and pooling features separately from each grid [57]. YOLOv3-spp performs better in handling objects of various sizes compared to YOLOv3. The SPP module in YOLOv3-spp enhances spatial reasoning and enables accurate detection of small objects.

The differences between YOLOv8n (Nano), YOLOv8l (Large), and YOLOv8x (Extra Large) lie in their model size, computational complexity, and accuracy. YOLOv8n is the smallest and fastest variant, designed for real-time applications on edge devices with limited processing power. YOLOv8l is a more balanced model, providing significantly better detection performance while still maintaining reasonable inference speeds. YOLOv8x, the largest model, maximizes accuracy, but it comes at the cost of slower processing speeds and higher computational demands [26].

The same happens for YOLOv11. The differences lies on the size of the network and the number os parameters used, following the same pattern as for YOLOv8 [27].

To achieve the objective of this paper, the three different databases created for this paper were used to train each of the different base weight files, resulting in 27 separate neural networks, all capable of detecting truck axles in truck images. Then, by a comparative analysis of the results, it was possible to evaluate the feasibility of employing synthetic images as training data for neural networks and assess their impact on the efficacy of truck axle detection.

4.3 Evaluation

The choice of performance metrics in truck axle detection depends on the relative costs associated with false positives and false negatives. When the costs attributed to false positive outcomes are substantial, 'precision' serves as a valuable comparison metric. On the other hand, if the costs associated with false negatives are higher, 'recall' emerges as a suitable metric. Misclassifying a truck axle as either a false positive or a false negative could lead to incorrect categorization, resulting in erroneous fees and weight limit determinations, for example. Given its ability to balance precision and recall, the 'F1-Score' assumes significance as an essential metric for evaluation and comparison purposes. Additionally, the 'Mean Average Precision' (mAP) metric is employed to assess the models' object detection capabilities, specifically discerning how they accurately localize the target area. Therefore, mAP was also included in the analytical assessment.

To evaluate the statistical significance of the results from all models, the Mann-Whitney U Test was applied. The Mann-Whitney test is suitable for comparing two independent samples, such as the mAP results of two distinct models, without assuming normality or homogeneity of variances [58]. Additionally, the Mann-Whitney test is robust to outliers and variations in data scale, which is particularly relevant in computer vision tasks, where results can be influenced by factors such as lighting, occlusion, or dataset variations. This robustness ensures that comparisons remain reliable even in scenarios where data exhibit high variability.

4.4 Experimental Results

The experiment involved utilizing 36 distinct images, different from the training set, and encompassing a total of 119 truck axles. As all the images were obtained from actual road traffic, they captured various wheel and hubcap configurations. The algorithms were developed to leverage the processing capabilities of the video board, specifically a GeForce GTX 1080. They were implemented using Python programming language, incorporating TensorFlow and Keras packages. Figure 9 shows an example of output detection images.

The resulting 27 neural network models were evaluated by comparing precision, recall, F1-score, and mAP. These metrics are used to measure the performance related to finding and classifying objects, which is the paper's main focus of comparison.

Figure 9: Results from the YOLOv3-spp network, with truck axle detections, trained on the three available databases (a) Real Trucks (b) Synthetic Trucks (c) Mixed Real and Synthetic Trucks.

5 Results and Discussions

The detection results, generated by training and applying all models to the testing dataset, can be seen in Table 3. The results only show the number of axles detected, with false positives and negatives. Based on those numbers, it is possible to notice that YOLOv3-tiny benefited the most from using the mixed image dataset with both synthetic and real trucks.

The other models presented consistent performance across all tests, indicating that integrating synthetic images effectively expands the diversity and quantity of available data for training purposes. This also suggests the potential of using synthetic images from a video game to enhance machine learning models' overall performance and robustness, providing valuable insights into the advancement of data augmentation techniques and their applicability in various domains.

The performance increased when using both images in the mixed database, reaching acceptable detection results for the smaller models. Detections were also better when analyzing the larger models, with increases in true positives and decreases in false positives.

The compromise in detection performance for YOLOv3-tiny resulted in many false positives and

Table 3: Detection results for truck axles on all tested neural networks.

Training Database	Model	True Positive	False Positive	False Negative
Real Trucks	YOLOv3-tiny	2	153	117
	YOLOv3	111	11	8
	YOLOv3-spp	117	2	2
	YOLOv8n	88	9	31
	YOLOv8l	112	6	7
	YOLOv8x	118	2	1
	YOLOv11n	91	4	28
	YOLOv11l	111	1	8
	YOLOv11x	118	1	1
Synthetic Trucks	YOLOv3-tiny	6	36	113
	YOLOv3	118	3	1
	YOLOv3-spp	117	1	2
	YOLOv8n	92	9	27
	YOLOv8l	112	5	7
	YOLOv8x	118	1	1
	YOLOv11n	92	3	27
	YOLOv11l	113	1	6
	YOLOv11x	118	1	1
Mixed Trucks	YOLOv3-tiny	92	9	27
	YOLOv3	116	2	3
	YOLOv3-spp	117	2	2
	YOLOv8n	99	7	20
	YOLOv8l	115	2	4
	YOLOv8x	119	1	0
	YOLOv11n	91	3	28
	YOLOv11l	114	1	5
	YOLOv11x	119	1	0

false negatives when using only real-world or synthetic trucks while training. Looking at the more recent variants of YOLO, the smaller models also had a worse performance when compared with the larger models. It is possible to see examples of false detections in Figure 10.

With true positives, false positives, and false negatives, it is possible to calculate the metrics used in this paper. The results can be seen in Table 4 and visualized in Figure 11, with graphs. Results are all on percentages, going from 0 to 100%. The perfect score is 100%, with all truck axles detected and in the correct position.

From the models trained with synthetic truck images, YOLOv3-tiny performed poorly with very low mAP (1.78%) and recall (5.04%), indicating many missed objects. Additionally, its precision and F1-score were relatively low. In contrast, the YOLOv3 and YOLOv3-spp models demonstrated excellent performance. They achieved high mAP values (99.06% and 98.30%, respectively) and recall values (99.16% and 98.32%, respectively). These models also had high precision and F1-scores. Like the models trained on synthetic images, YOLOv3 and YOLOv3-spp trained on mixed images performed well. They had high mAP values (97.46% and 98.28%, respectively) and recall values (97.48% and 98.32%, respectively). These models also demonstrated high precision and F1-scores.

On the other hand, the YOLOv3-tiny model trained on real-world trucks had the worst performance among all the models. It had extremely low mAP (0.03%), recall (1.68%), and precision (1.29%) values. This model missed a significant number of objects and had a high number of false positives. The YOLOv3 and YOLOv3-spp models trained on real-world trucks had relatively high mAP values (92.01% and 98.26%, respectively) and good recall values (93.28% and 98.32%, respectively). Although they performed well, their performance did not match the models trained on synthetic and mixed truck images.

YOLOv8 followed a similar pattern, with YOLOv8n having a worse performance compared to

Figure 10: False negatives, when truck axles are not detected, and false positives, when objects that are not axles are detected and classified as axles.

YOLOv8l and YOLOv8x. It is important to notice that although results were indeed worse, they were still better when compared with YOLOv3tiny. This indicates that the evolution of the smaller models, with reduced number of layers and parameters, are still having an influence on the results, but by a diminishing factor. That's an interesting result, given that smaller models are capable of running on

Table 4: Calculated metrics for all trained models.

Training Database	Model	Recall (%)	Precision (%)	F1-score (%)	mAP (%)
Real Trucks	YOLOv3-tiny	1.68	1.29	1.46	0.03
	YOLOv3	93.28	90.98	92.12	92.01
	YOLOv3-spp	98.32	98.32	98.32	98.26
	YOLOv8n	73.95	90.72	81.48	78.32
	YOLOv8l	94.12	94.92	94.51	92.06
	YOLOv8x	99.16	98.33	98.74	97.60
	YOLOv11n	76.47	95.79	85.05	89.98
	YOLOv11l	93.28	99.11	96.10	97.67
	YOLOv11x	99.16	99.16	99.16	98.88
Synthetic Trucks	YOLOv3-tiny	5.04	14.29	7.45	1.78
	YOLOv3	99.16	97.52	98.33	99.06
	YOLOv3-spp	98.32	99.15	98.73	98.30
	YOLOv8n	77.31	91.09	83.64	77.90
	YOLOv8l	94.12	95.73	94.92	92.15
	YOLOv8x	99.16	99.16	99.16	98.49
	YOLOv11n	77.31	96.84	85.98	82.65
	YOLOv11l	94.96	99.12	97.00	98.12
	YOLOv11x	99.16	99.16	99.16	99.02
Mixed Trucks	YOLOv3-tiny	77.31	91.09	83.64	76.42
	YOLOv3	97.48	98.31	97.89	97.46
	YOLOv3-spp	98.32	98.32	98.32	98.28
	YOLOv8n	83.19	93.40	88.00	78.59
	YOLOv8l	96.64	98.29	97.46	92.09
	YOLOv8x	100.00	99.17	99.58	98.57
	YOLOv11n	76.47	96.81	85.54	81.20
	YOLOv11l	95.80	99.13	97.44	98.19
	YOLOv11x	100.00	99.17	99.58	99.04

portable devices, contributing to the process of 'IoTzation'.

The same can be said about YOLOv11. All models followed the same pattern of YOLOv8, with an improve in detection results, reaching 100% recall on the mixed database. Looking at the other metrics, all results on the synthetic and mixed database were above 99%, indicating an improvement on performance, and giving a hint about the overall result of this paper.

For all trained models, this paper demonstrated that using a mixed database with synthetic and real-world images meant better or at least comparable results for all trained models. This result might be due to the variability of examples to extract information from during the training phase, which the real-world database lacks. All trucks on the real-world database were from a side viewpoint, perpendicular to the road. Synthetic images have a more comprehensive range of axle positions, as shown in Figure 12.

(a)

(b) (c)

Figure 12: Variable positions of truck axles in the synthetic image database, providing more information for the neural networks to extract patterns from during the training phase.

The Mann-Whitney U test was applied to the resulting mAP values to determine whether the differences between them are statistically significant or not. Two null hypothesis were constructed and tested.

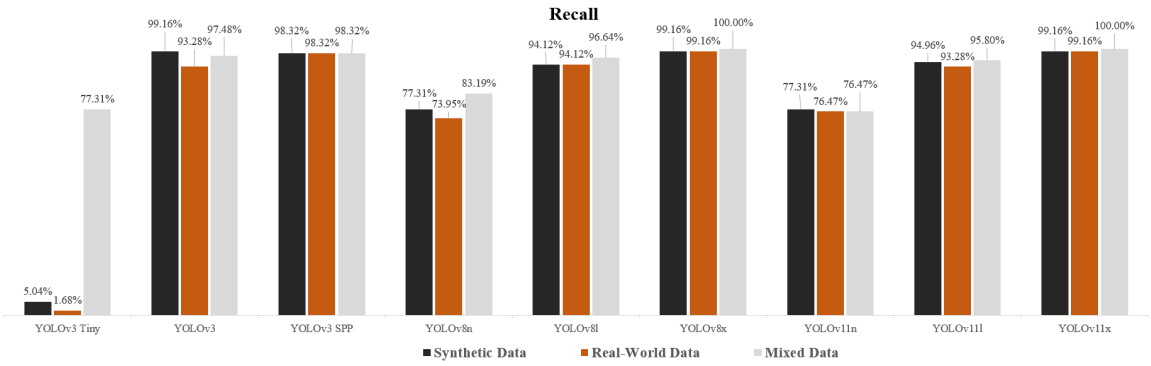
The first hypothesis, 'H_0 : There are no differences between neural networks trained on different databases of images.' was created to check if there are any difference between mAp results of neural networks trained on different image databases.

Since there are three possible comparisons to be made, Real Data x Synthetic Data, Real Data x Mixed Data, and Synthetic Data x Mixed Data, the test was applied to each one, with a resulting U value. The U value was then checked against the critical value table, and results are shown on Table 5.

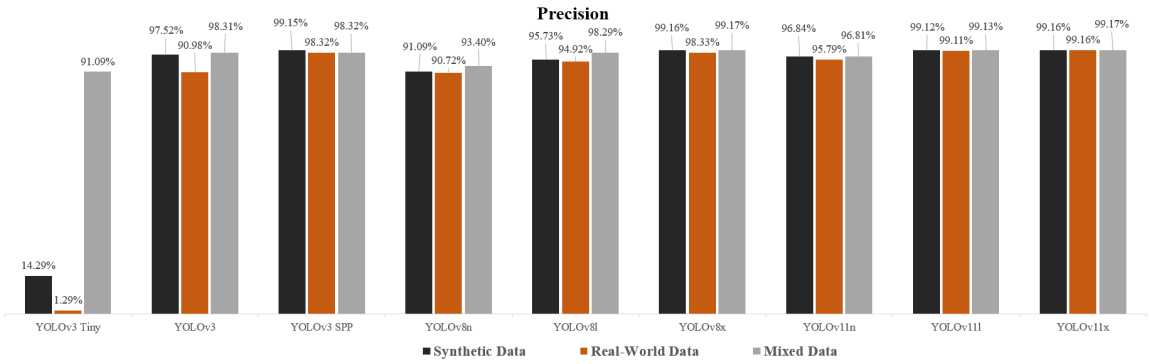
Table 5: Results from the Mann-Whitney U test for the three image databases.

	U Value	Critical Value*	Results
Real Data x Synthetic Data	31	17	Fail to Reject
Real Data x Mixed Data	34	17	Fail to Reject
Synthetic Data x Mixed Data	44	17	Fail to Reject

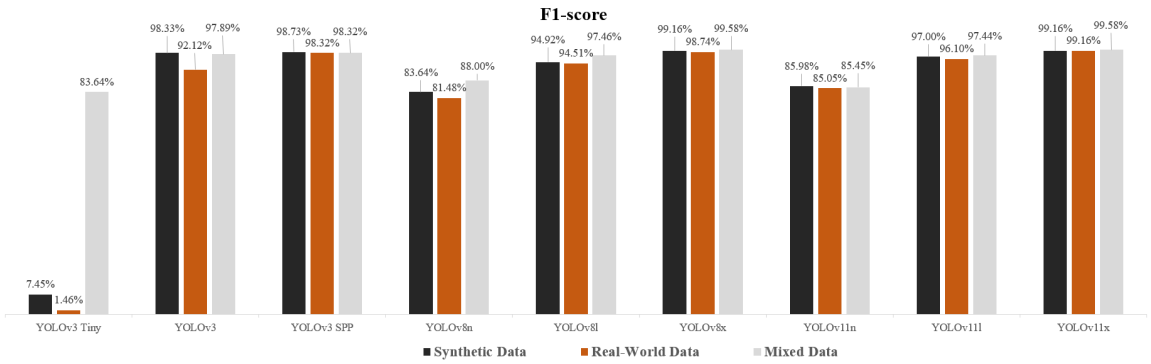
*$\alpha = 0.05, n1 = 9, n2 = 9$



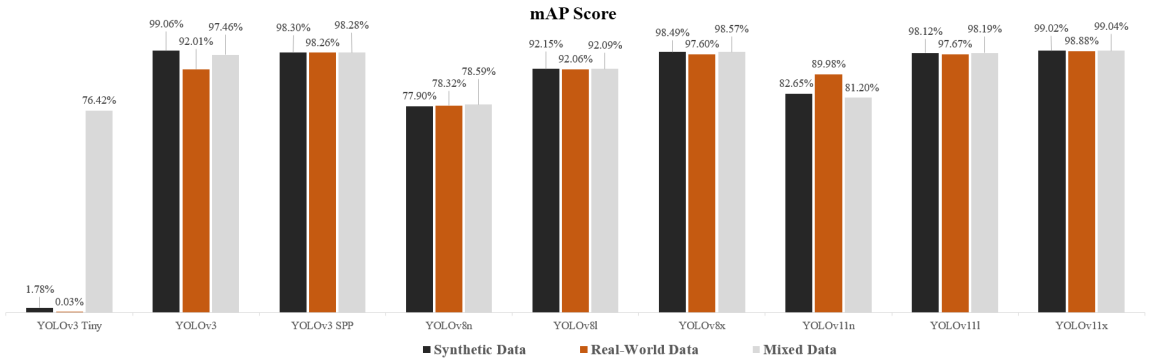
(a)



(b)



(c)



(d)

Figure 11: Graphs comparing all metrics calculated on this paper, in percentages (a) Recall (b) Precision (c) F1-score (d) mAP.

Since the critical value for all tests is 17, we fail to reject the null hypothesis. Therefore, the mAp results distribution from all models are equivalent between each other. This indicates that it is possible to train neural networks using synthetic images without losing performance.

It is possible to see the p-values of the Mann-Whitney U tests on Figure 13. Results are significant, with p-values above 0.43.

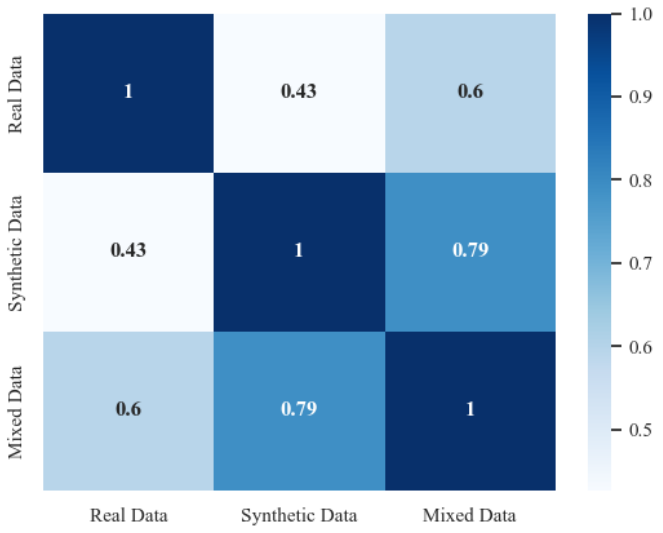


Figure 13: p-values from the Mann-Whitney U test between mAP results from different databases.

The second hypothesis tested, 'H_0 : There are no differences on mAP results between versions of YOLO neural networks when trained on different sets of images.' was created to check whether results from the different versions of the YOLO architecture were significant or not on the databases created for this paper.

Three comparisons are also possible between the versions of the YOLO neural network used on this paper, YOLOv3 x YOLOv8, YOLOv3 x YOLOv11, and YOLOv8 x YOLOv11. All comparisons were also tested with the U test, and results are seen on Table 6.

With the same resulting critical value of 17, it is not possible to reject the null hypothesis, indicating that the mAP performance between model versions is equivalent. This happens due to the high detection levels of all trained models, since mAp scores were usually above 90%.

The p-values can be seen on Figure 14, with the lowest p-value of 0.16 on the comparison between YOLOv8 and YOLOv11.

Based on all results, it is possible to conclude that using synthetic images from a video game to train neural networks is a viable option, contributing with different examples for extracting knowledge and enriching the dataset. The trained models either improved detection performance or got similar results to training with real-world images.

Furthermore, extracting the images from the video game used less time, financial, and human resources, with no need to go to the roadside to record trucks. By opting for synthetic images, the risks

Table 6: Results from the Mann-Whitney U test for the three versions of YOLO.

	U Value	Critical Value*	Results
YOLOv3 x YOLOv8	39	17	Fail to Reject
YOLOv3 x YOLOv11	33	17	Fail to Reject
YOLOv8 x YOLOv11	24	17	Fail to Reject

*$\alpha = 0.05, n1 = 9, n2 = 9$

Figure 14: p-values from the Mann-Whitney U test between mAP results from different YOLO versions.

associated with standing beside a highway, vulnerable to vehicular accidents, exhaust fumes or adverse weather, are effectively negated. Video game images offer the advantage of customization, allowing scenarios to be crafted, increasing variety in examples for the neural network. Thus, prioritizing the integration of video game imagery not only safeguards researchers from potential harm but also ensures the creation of content in a controlled environment.

6 Conclusion and Future Works

We tested three different databases, with real-world truck images, synthetic truck images, and mixed images, to train various YOLO networks. The training process took advantage of already learned knowledge of nine base weight files for three distinct sized networks, from smaller to larger: (1) YOLOv3-tiny, YOLOv8n and YOLOv11n, (2) YOLOv3, YOLOv8l, YOLOv11l and (3) YOLOv3-spp, YOLOv8x, YOLOv11x. We evaluated the 27 resulting models using recall, precision, f1-score, and mAP to assess if training a neural network using synthetic images from a video game is possible. Results show that it is possible, with all models performing better or equivalent to neural networks trained with real-world images. This finding is the main contribution of this paper since it opens the possibility to use synthetic images from a video game to enhance the number of examples to train a neural network, saving time and resources.

Mixing images resulted in a better amount of variability, with truck axles in several different angles and lighting conditions. Since neural networks extract knowledge from all examples provided, synthetic images are an excellent choice to enrich a database of images, especially if they come from a video game with realistic graphics. The closer the representation of the object in the synthetic image to real life, the better the results will be.

The results of this paper also showed that even with smaller databases for training and testing, detection and classification were successful. This suggests that using a specialized neural network, specifically trained for a particular task, is a viable alternative to using pre-trained, general-purpose neural networks found online. Moreover, not only is it a good alternative, but it can also be trained with small databases. This is a significant finding because acquiring sufficient data can be challenging for researchers. Therefore, this paper highlights the possibility of using small databases to train specialized neural networks with good results.

To ensure a fair comparison of results, training and testing all networks using the same databases was crucial. Although the training times for each model were significantly longer than the testing times, that was not considered since training is performed only once for each network. Each model required

approximately four hours of training, while testing was completed in seconds. While better hardware can reduce training times, larger databases require more processing power for faster processing times. Given that real-world images may be mixed with easily acquired synthetic images, training databases will get larger in future works.

7 Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. This research also received financial support from the National Council for Scientific and Technological Development (CNPq), projects 436954/2018-4 and 311964/2022-2.

References

[1] O. Avci, O. Abdeljaber, and S. Kiranyaz, "An overview of deep learning methods used in vibration-based damage detection in civil engineering," *Dynamics of Civil Structures, Volume 2*, pp. 93–98, 2022.

[2] X. Gan, J. Pei, W. Wang, S. Yuan, and B. Lin, "Application of a modified mopso algorithm and multi-layer artificial neural network in centrifugal pump optimization," *Engineering Optimization*, vol. 55, no. 4, pp. 580–598, 2023.

[3] S. Scholes, A. Ruget, G. Mora-Martín, F. Zhu, I. Gyongy, and J. Leach, "Dronesense: The identification, segmentation, and orientation detection of drones via neural networks," *IEEE Access*, vol. 10, pp. 38154–38164, 2022.

[4] D. Çelebi, B. Bolat, and D. Bayraktar, "Light rail passenger demand forecasting by artificial neural networks," *2009 International Conference on Computers & Industrial Engineering*, pp. 239–243, 2009.

[5] A. Badr, M. M. Abdelwahab, A. Thabet, and A. Abdelsadek, "Automatic number plate recognition system," *International Research Journal of Modernization in Engineering Technology and Science*, 2011.

[6] A. A. Chernyshev, E. A. Koryagina, D. G. Moroz, and S. S. Titova, "The use of artificial neural networks (ann) as an auxiliary factor in planning transportation routes : Theoretical aspects of artificial intelligence systems development for transportation engineering," *2022 Systems of Signals Generating and Processing in the Field of on Board Communications*, pp. 1–4, 2022.

[7] S. Mehtab and W. Q. Yan, "Flexible neural network for fast and accurate road scene perception," *Multimedia Tools and Applications*, vol. 81, no. 5, pp. 7169–7181, 2022.

[8] L. A. Marcomini and A. L. Cunha, "Truck axle detection with convolutional neural networks," 2023.

[9] P. Hensman and D. Masko, "The impact of imbalanced training data for convolutional neural networks," *Degree Project in Computer Science, KTH Royal Institute of Technology*, 2015.

[10] C. Zhang, E. Nateghinia, L. F. Miranda-Moreno, and L. Sun, "Pavement distress detection using convolutional neural network (cnn): A case study in montreal, canada," *International Journal of Transportation Science and Technology*, vol. 11, no. 2, pp. 298–309, 2022.

[11] S. A. Coleman, D. Kerr, and Y. Zhang, "Image sensing and processing with convolutional neural networks," *Sensors (Basel, Switzerland)*, vol. 22, 2022.

[12] Y. Roh, G. Heo, and S. E. Whang, "A survey on data collection for machine learning: a big data – ai integration perspective," 2019.

[13] M. Sajjad, S. Khan, K. Muhammad, W. Wu, A. Ullah, and S. W. Baik, "Multi-grade brain tumor classification using deep cnn with extensive data augmentation," *Journal of computational science*, vol. 30, pp. 174–182, 2019.

[14] Y. Yan, Z. Tan, and N. Su, "A data augmentation strategy based on simulated samples for ship detection in rgb remote sensing images," *ISPRS International Journal of Geo-Information*, vol. 8, no. 6, p. 276, 2019.

[15] D. Bhatt, C. Patel, H. Talsania, J. Patel, R. Vaghela, S. Pandya, K. Modi, and H. Ghayvat, "Cnn variants for computer vision: history, architecture, application, challenges and future scope," *Electronics*, vol. 10, no. 20, p. 2470, 2021.

[16] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects," *IEEE transactions on neural networks and learning systems*, 2021.

[17] S. Balaji, "Binary image classifier cnn using tensorflow." https://medium.com/techiepedia/binary-image-classifier-cnn-using-tensorflow-a3f5d6746697, Aug. 2020. Accessed: 2023-06-01.

[18] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[21] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[23] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[24] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[25] A. Ammar, A. Koubaa, M. Ahmed, A. Saad, and B. Benjdira, "Vehicle detection from aerial images using deep learning: A comparative study," *Electronics*, vol. 10, no. 7, p. 820, 2021.

[26] Ultralytics, "Yolov8 - ultralytics yolo docs." https://docs.ultralytics.com/pt/models/yolov8/. Accessed: 2025-02-26.

[27] Ultralytics, "Yolo11 - ultralytics yolo docs." https://docs.ultralytics.com/pt/models/yolo11/. Accessed: 2025-02-26.

[28] R. Khanam and M. Hussain, "Yolov11: An overview of the key architectural enhancements," *arXiv preprint arXiv:2410.17725*, 2024.

[29] S. Coleman, D. Kerr, and Y. Zhang, "Image sensing and processing with convolutional neural networks," 2022.

[30] M. G. Pauly, M. Ruiz Lopez, A. Westenberger, G. Saranza, N. Brüggemann, A. Weissbach, R. L. Rosales, C. C. Diesta, R. D. Jamora, C. J. Reyes, *et al.*, "Expanding data collection for the mdsgene database: X-linked dystonia-parkinsonism as use case example," *Movement Disorders*, vol. 35, no. 11, pp. 1933–1938, 2020.

[31] V. N. Gulin, A. A. Neretin, and S. Paudyal, "Bim of transport infrastructure – practical aspects of data collection for dtm creation," *IOP Conference Series: Materials Science and Engineering*, vol. 832, 2020.

[32] V. Macchiarulo, P. Milillo, C. E. Blenkinsopp, C. Reale, and G. Giardina, "Multi-temporal insar for transport infrastructure monitoring: Recent trends and challenges," *Proceedings of the Institution of Civil Engineers - Bridge Engineering*, 2022.

[33] T. V. Louro, J. U. P. Junior, G. T. d. O. Gardin, and C. A. P. da Silva Junior, "Factors influencing lateral distance and speed of motorized vehicles overtaking bicycles," *Transportation Research Record*, p. 03611981221150926, 2023.

[34] A. B. Morelli, A. d. C. Fiedler, and A. L. Cunha, "Um banco de dados de empregos formais georreferenciados em cidades brasileiras," *arXiv preprint arXiv:2303.09602*, 2023.

[35] N. Safaei, O. Smadi, A. Masoud, and B. Safaei, "An automatic image processing algorithm based on crack pixel density for pavement crack detection and classification," *International Journal of Pavement Research and Technology*, vol. 15, no. 1, pp. 159–172, 2022.

[36] E. V. Butilă and R. G. Boboc, "Urban traffic monitoring and analysis using unmanned aerial vehicles (uavs): A systematic literature review," *Remote Sensing*, vol. 14, no. 3, p. 620, 2022.

[37] X. Dong, S. Yan, and C. Duan, "A lightweight vehicles detection network model based on yolov5," *Engineering Applications of Artificial Intelligence*, vol. 113, p. 104914, 2022.

[38] S. E. Whang, Y. Roh, H. Song, and J.-G. Lee, "Data collection and quality challenges in deep learning: A data-centric ai perspective," *The VLDB Journal*, pp. 1–23, 2023.

[39] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," 2018.

[40] N. Arora and Y. Kumar, "Automatic vehicle detection system in day and night mode: challenges, applications and panoramic review," *Evolutionary Intelligence*, pp. 1–19, 2022.

[41] S. A. Assefa, D. Dervovic, M. Mahfouz, R. E. Tillman, P. Reddy, and M. Veloso, "Generating synthetic data in finance: opportunities, challenges and pitfalls," in *Proceedings of the First ACM International Conference on AI in Finance*, pp. 1–8, 2020.

[42] V. Seib, B. Lange, and S. Wirtz, "Mixing real and synthetic data to enhance neural network training – a review of current approaches," 2020.

[43] Q. Wang, J. Gao, W. Lin, and Y. Yuan, "Learning from synthetic data for crowd counting in the wild," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8198–8207, 2019.

[44] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Synthetic data and artificial neural networks for natural scene text recognition," 2014.

[45] K. W. Dunn, C. Fu, D. J. Ho, S. Lee, S. Han, P. Salama, and E. J. Delp, "Deepsynth: Three-dimensional nuclear segmentation of biological images using neural networks trained with synthetic data," *Scientific reports*, vol. 9, no. 1, pp. 1–15, 2019.

[46] T. Björklund, A. Fiandrotti, M. Annarumma, G. Francini, and E. Magli, "Automatic license plate recognition with convolutional neural networks trained on synthetic data," in *2017 IEEE 19th international workshop on multimedia signal processing (MMSP)*, pp. 1–6, IEEE, 2017.

[47] S. Goswami, "Impact of data quality on deep neural network training," 2020.

[48] P. Ventura Diaz and S. Yoon, "High-fidelity simulations of a quadrotor vehicle for urban air mobility," in *AIAA SCITECH 2022 Forum*, p. 0152, 2022.

[49] C. I. de Campos, L. A. Marcomini, N. R. Panice, F. J. Piva, and A. P. C. Larocca, "Perception analysis of highway quality of service using a driving simulator and eye tracking system," *Transportes*, vol. 28, no. 3, pp. 165–179, 2020.

[50] A. Rizwan, D. A. Karras, M. Dighriri, J. Kumar, E. Dixit, A. Jalali, and A. Mahmoud, "Simulation of iot-based vehicular ad hoc networks (vanets) for smart traffic management systems," *Wireless Communications and Mobile Computing*, vol. 2022, 2022.

[51] G. B. A. Wicaksana and M. A. W. Linggasani, "Model into playable simulation in games cities," in *International Webinar on Digital Architecture 2021 (IWEDA 2021)*, pp. 302–306, Atlantis Press, 2022.

[52] P. R. Wurman, S. Barrett, K. Kawamoto, J. MacGlashan, K. Subramanian, T. J. Walsh, R. Capobianco, A. Devlic, F. Eckert, F. Fuchs, *et al.*, "Outracing champion gran turismo drivers with deep reinforcement learning," *Nature*, vol. 602, no. 7896, pp. 223–228, 2022.

[53] S. Software, "Euro truck simulator 2." https://eurotrucksimulator2.com/. Accessed: 2023-06-22.

[54] S. Software, "World of trucks." https://www.worldoftrucks.com/en/. Accessed: 2023-06-05.

[55] S.-U. Maya-Martínez, A.-J. Argüelles-Cruz, Z.-J. Guzmán-Zavaleta, and M.-d.-J. Ramírez-Cadena, "Pedestrian detection model based on tiny-yolov3 architecture for wearable devices to visually impaired assistance," *Frontiers in robotics and AI*, vol. 10, p. 1052509, 2023.

[56] Z. Yi, S. Yongliang, and Z. Jun, "An improved tiny-yolov3 pedestrian detection algorithm," *Optik*, vol. 183, pp. 17–23, 2019.

[57] Z. Huang, J. Wang, X. Fu, T. Yu, Y. Guo, and R. Wang, "DC-SPP-YOLO: Dense connection and spatial pyramid pooling based YOLO for object detection," *Information Sciences*, vol. 522, pp. 241–258, jun 2020.

[58] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, pp. 50–60, 1947.