

# Automatic detection of scratching events on vehicles with audio-based spectrograms

André R. Soares <sup>a,b</sup>, André L. Ferreira <sup>a,b</sup> , João M. Fernandes <sup>b,c</sup> ,\*

<sup>a</sup> Bosch Car Multimedia Portugal S.A., Braga, Portugal

<sup>b</sup> Centro ALGORITMI/Dep. Informática, Universidade do Minho, Braga, Portugal

<sup>c</sup> CCG/ZGDV, Guimarães, Portugal

## ARTICLE INFO

### Keywords:

Deep learning  
Scratch  
Damage detection  
Convolutional neural network  
Automotive

## ABSTRACT

The identification of damages, specifically scratches, on the structure of vehicles is a challenge to many businesses. In most automobile industries, this process is mainly performed by human vision, which is unstable and error-prone. Companies are avidly seeking for automatic solutions to facilitate this type of inspection. This article is about an endeavour that intends to develop a novel software-based solution for automatically identifying events that can cause scratches on passenger vehicles. The algorithm that identifies scratching events is at the heart of that solution. The algorithm is based on a convolutional neural network and was designed to use audio data obtained from microphones installed in vehicles.

The article presents methods for transforming audio signals into images representing the spectral characteristics of the audio. It also describes the development and the evaluation of the convolutional neural network that was trained to identify scratching events. The convolutional neural network undergoes training, considering the vast array of sounds that a microphone can detect, within the countless everyday driving environments which vehicles can be subjected to.

The accuracy of the convolutional neural network that identifies scratching events has an excellent performance, as indicated by a Matthews correlation coefficient equal to +0.90. Our work highlights the potential of convolutional neural networks in scratching events identification and prepares for future research to expand the dataset and to incorporate a wider range of events to be detected. These promising results permit one to consider the use of the algorithm in different applications for the automotive domain.

## 1. Introduction

Artificial Intelligence undeniably has a significant impact on society across various domains. Deep Learning (DL), in particular, has revolutionised the way machines learn by employing efficient techniques to tackle complex problems, driving innovation in numerous industries that benefit from this technology. One such industry is the automotive industry, where there have been major changes in ownership models. Nowadays, many persons are looking for viable alternatives to vehicle ownership, particularly in large cities with well-developed public transportation infrastructure. Choosing to rent a vehicle for occasional trips eliminates concerns related to own a vehicle, like maintenance costs, taxes, and insurance, providing a fast and accessible way of getting around a city. Renting a vehicle is an attractive option for a large number of persons, that put aside the otherwise usual demand of owning one.

From a business standpoint, it is evident that customers do not always take the best care of the rental vehicle while they are in its

full possession. Therefore, a comprehensive inspection for damages at the end of each rental period becomes necessary. However, relying solely on human inspection can be both laborious and prone to errors, potentially resulting in financial losses for the company due to overlooked damages. Given this context, it becomes crucial to monitor both the internal and external surfaces/parts of the vehicle, equipping it with sensory capabilities to detect impacts that may lead to damages (scratches, dents, etc.). However, it is a challenge to inspect such defects in a computer system, due to the imbalanced illumination, specular highlight reflection, various reflection modes and limited defect features.

Bosch is currently working on a product for a car group. The company developed an anomaly audio detection task for the automotive domain, where image-like time–frequency representations of audio are explored (Coelho et al., 2022). A CNN-based algorithm was used with success in detecting abnormal sounds inside a vehicle cabin. This article

\* Corresponding author at: Centro ALGORITMI/Dep. Informática, Universidade do Minho, Braga, Portugal.

E-mail addresses: [a67654@alunos.uminho.pt](mailto:a67654@alunos.uminho.pt) (A.R. Soares), [Andre.Ferreira2@pt.bosch.com](mailto:Andre.Ferreira2@pt.bosch.com) (A.L. Ferreira), [jmf@di.uminho.pt](mailto:jmf@di.uminho.pt) (J.M. Fernandes).

is a continuation of that previous work. The algorithm presented in this article is part of system to be use in a car-sharing service available to the general public on demand, aiming to provide a more user-friendly and efficient system. The system aims to solve the damage inspection problem by leveraging CNN-based algorithms to identify scratches in a vehicle equipped with sensory capabilities, by gathering audio data with microphones. So, instead of trying to identify the scratches directly in the vehicle surfaces, the product aims to identify, using microphones, the events that have potential to create scratches in the vehicle. This product can have a number of different applications in the automotive industry. It promises to significantly reduce inspection costs, to mitigate human errors in damage assessments, to offer valuable insights about the condition of the vehicle to the driver, and to provide real-time updates to car rental companies and to insurance companies regarding the status of vehicles.

### 1.1. Major aim and objectives

The major aim of this work is **to develop an CNN-based algorithm to detect scratches in vehicles, using audio-data (obtained with microphones)**. This is achieved through the transformation of audio data into a visual format (spectrograms), coupled with the deployment of advanced CNN techniques for classification and detection.

In pursuit of the major aim, this work is guided by two objectives:

- O1: To develop a method for scratching events detection in vehicles through the transformation of audio signals using a CNN. This step aims to check the feasibility of detecting scratches by using images that are visual representations of the signal when these are performed in a controlled environment. This means ideal situations where the scratching event occurs without background (non-scratching) noises.
- O2: To develop a method for scratching events detection in vehicles that can handle more realistic scenarios by accounting for the polyphonic acoustic environment to which a vehicle is exposed.

Throughout the development process, the emphasis was placed on assessing the impact of various techniques, such as data augmentation, transfer learning, and types of visual representations, on the overall accuracy and effectiveness of the models. The CNN-based algorithm must be reliable and it is expected to achieve a high Matthews correlation coefficient (MCC), making its use feasible and valuable for implementation in industrial applications.

### 1.2. Key contributions

To our knowledge, this is the first approach reported in the literature developed to detect scratches in vehicles based on audio-based spectrograms for industrial purposes. The following are the key contributions of this work:

- Description of the data collection, model selection, and the utilisation of transfer learning and fine tuning techniques to build a robust detection algorithm for scratching events in vehicles.
- Organisation of a training set based on a series of experiments conducted on audio and images.
- Use of log-mel spectrograms as the optimal choice for scratching events detection with CNNs.
- Use of the VGG16 CNN model for conducting all the development effort.
- The CNN-based algorithm achieved MCC scores of over 0.90 for the best models, which are considered excellent and well-suited for many industrial applications.

The remaining part of the paper is organised as follows. Section 2 describes the related work on CNNs and on the use of audio-based spectrograms in scratch identification, namely in vehicles. Section 3 presents the steps that conducted to the tuning of the scratch detection algorithm. Section 4 presents the selected CNN model and how it was parameterised to cope with the problem under consideration. Section provides the discussion and performance analysis. The conclusions and future work are presented in Section 7.

## 2. Related work

This section discusses two important topics within this manuscript: (1) CNNs, and (2) audio-based spectrograms for scratch identification.

### 2.1. Convolutional neural networks

Convolutional neural network (CNNs) are loosely based on the manner in which all mammals perceive the world around them, using a hierarchical series of feature recognition mechanisms. This process starts with simpler features like diagonal lines or curved edges and progresses towards more complex and abstract recognitions, such as combinations of shapes and finally the classification of entire objects. CNNs have emerged as the gold standard for image-related tasks, specifically designed to process data represented as arrays. For instance, in the case of a colour image or an audio spectrogram, the computer perceives the image as a matrix of pixels with height, width, and depth, where each colour channel (RGB) corresponds to a 2D array. Therefore, a  $24 \times 24 \times 3$  image matrix refers to a  $24 \times 24$  image size with three colour channels. If we are processing an audio signal or sequences 1D arrays would be used or 3D arrays in the case of volumetric images. Deep CNNs have made significant breakthroughs in image, video, speech, and audio processing, successfully applied to tasks such as object detection, segmentation, and recognition of objects in images such as traffic sign recognition (Shangzheng, 2019), segmentation of biological features (Perry & Fernandez, 2019), and face detection (Qu et al., 2018).

The architecture of a CNN consists of two fundamental types of layers: convolutional layers and pooling layers (LeCun, Bengio, & Hinton, 2015).

- **Convolutional layer:** This layer constitutes the basic unit of a CNN and it is where one can find most of the computational part. It applies convolution to the input, generating results that are subsequently passed on to the next layer of the neural network (NN). The convolution operation involves applying a filter to the entire image, extracting relevant features. Fig. 1(a) shows an example of a convolution operation. By applying the kernel to the small green patch in the input matrix, the result obtained with  $4 \times 0 + 3 \times 1 + 0 \times 2 + 1 \times 3$  that equals 6 for the first element in the output matrix.
- **Pooling Layer:** Downsampling is employed to efficiently reduce the dimensionality of the data representation, enabling the CNN to utilise fewer parameters. This makes the CNN more computationally efficient, particularly during training. Pooling involves selecting a filter size and a stride, moving the filter across the image and applying the pooling operation. Common pooling operations include max pooling, which returns the maximum value captured by the window applied throughout the image, and average pooling, which returns the average value. Pooling also helps address the issue of overfitting (Aloysius & Geetha, 2017). Fig. 1(b) shows an example of the max pooling operation applied to a matrix, where the maximum value, captured by a  $2 \times 2$  filter with strides of  $2 \times 2$ , is passed to the output matrix.

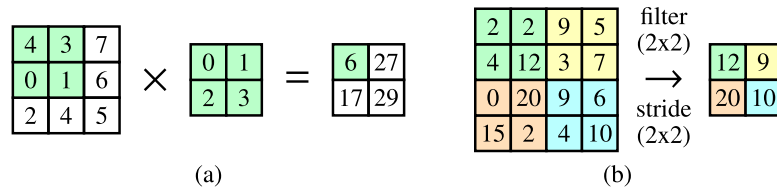


Fig. 1. (a) Convolution and (b) max pooling operations.

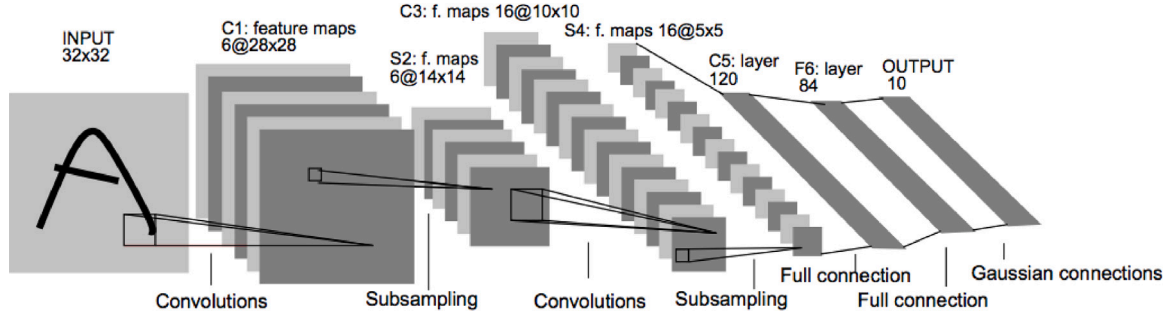


Fig. 2. The architecture of LeNet5 (Lecun et al., 1998).

After the feature extraction phase in the convolutional and pooling layers, the CNN utilises fully-connected layers. These layers are simple feed-forward NNs layers, where each node is connected to every node in the previous and next layer. Fully-connected layers perform classification tasks. However, one major drawback of fully-connected layers is their large number of parameters, which increases the likelihood of neuron interdependence (i.e., overfitting). To mitigate this, regularisation techniques such as dropout are often implemented at this stage.

The architecture of a CNN plays a crucial role in determining its performance and efficiency. The arrangement of layers, the components within each layer, and the overall design have a significant impact on the speed and accuracy of the CNN in various tasks. Common CNN architectures often consist of a series of alternating convolutional and pooling layers, followed by fully connected layers. Many of these architectures have gained widespread usage and recognition, achieving state-of-the-art results in influential computer vision competitions, like the ImageNet Large Scale Visual Recognition Challenge (ILSVRC).

LeNet5, created by Lecun, Bottou, Bengio, and Haffner (1998) for handwriting recognition, is one of the pioneering CNN architectures. As illustrated in Fig. 2, it has seven levels.

CNNs have been used in various applications, including segmentation of indoor scene images (Zhu et al., 2021), detection of defects of leather (Iqbal, Khan, Naqvi, & Holmes, 2023), the Alzheimer's disease (Erdogmus & Kabakus, 2023), classification of rice varieties (Din et al., 2024), diagnosis of thunderstorm prediction (Jardines et al., 2024), and estimation of hand gestures (Shanmugam & Narayanan, 2024).

## 2.2. Audio-based spectrograms in scratch identification

As far as we are aware, there are not any scientific publications discussing the use of audio-based spectrograms in automotive scratch identification. There are however some proposals to detect defects on the exterior of vehicles based on different approaches.

Pachón-Suescún, Pinzón-Arenas, and Jiménez-Moreno (2019) present an algorithm based on convolutional neural networks and a variant of these using regions (CNN and R-CNN) that detect scratches in a vehicle. The algorithm uses the capture of one of the sides of a vehicle and an R-CNN extracts the region of the image where the vehicle is located. The extracted region is divided into various sections, and each

one is evaluated by a CNN to detect in which parts the scratches are located. With the test images, a precision percentage of 98.3% was obtained in the R-CNN, and 96.9% in the CNN.

van Ruitenbeek and Bhulai (2022) present a damage detection model to locate vehicle damages and classify the damages into twelve categories. They include scratches as a damage category and use various deep learning algorithms. Their final model is able to accurately detect small damages from 2D images under various conditions such as water and dirt.

Zhou et al. (2019) present the development of an automatic inspection system for automobile surface defects that are located in or close to style lines, edges and handles. Experimental results demonstrate that their system can effectively reduce false detection of pseudo-defects produced by image noise and achieve accuracies of 95.6% in dent defects and 97.1% in scratch defects.

Rao and Desai (2022) describe an automatic dent detection system that uses a Mask-R CNN algorithm. They discuss how to detect dents using infrared (IR) sensors for distance measurement from defined threshold. The authors indicate that the usage of IR sensors makes their solution unique, since those sensors are more reliable than any other sensor.

Arnal, Solanes, Molina, and Tornero (2017) introduce a new approach to detect defects, cataloged as dings and dents, on vehicle body surfaces, which is currently an important issue in the automotive industry. Their method consists of two major steps. In the first step, light patterns projected on the body surface sweep uniformly the area of inspection in the pre-processing step. Additionally, a new image fusion law based on optical flow is used to obtain a resulting fused image holding the information of all variations suffered by the projected patterns during the sweeping process, indicating the presence of anomalies. In the second step, a new post-processing step is proposed that eliminates the need of using pre-computed reference backgrounds to distinguish defects from other body features such as style-lines.

## 3. The scratch detection algorithm

In this section, we delve into the practical work, discussing key aspects such as data collection, model selection, and the utilisation of transfer learning and fine tuning techniques to build a robust vehicle scratch detection algorithm. DL has rapidly emerged as the state-of-the-art solution for computer vision problems. The objective of this article is to document the development of a vehicle scratch detection algorithm by leveraging audio captured inside a vehicle. The initial

phase involved extensive research to identify the best approaches for similar problems followed by careful reflection and strategy outlining to determine suitable techniques for the given situation. Subsequently, the machine learning (ML) cycle project was initiated, which encompasses a series of well-defined steps commonly employed by ML development teams. The project begins with data-related tasks, such as data collection, understanding the data, and preparing it for further processing. Then, a practical approach is taken, involving the selection of an appropriate model, training it on the prepared data, evaluating its performance, and fine-tuning hyperparameters. In this context, spectrograms serve as representations that depict the strength of audio signals over time and various frequencies, presented as images. These spectrogram images are commonly used with deep NNs to address image detection and classification tasks in computer vision. Right away, it becomes apparent that one fundamental difficulty with the challenge will be the distinction between the strength of the sound produced by scratches and the background noise present in the audio. In real-life scenarios, vehicles are subject to a wide variety of sounds originating from the infinitely possible environments where someone could drive to. Factors like speech, music, horns and engine noises can induce difficulties in scratch detection, but it is something that the algorithm should be ready to deal with.

This article documents a series of experiments conducted on audio and images, which were organised into a training set. The goal is to achieve results that are applicable to real-life implementations of scratch detection algorithms in vehicles. All the development work was carried out using an Anaconda environment for Python, utilising the latest version of TensorFlow along with the Keras API, which provides comprehensive support for every step of the ML workflow. Scikit-learn was employed for predictive data analysis, Numpy for efficient array manipulation and linear algebra operations, and Matplotlib for creating interactive visualisations to facilitate analysis within the project.

### 3.1. Transfer learning and fine tuning

To develop the system, the chosen approach was to leverage existing open-source CNN architectures using a prevalent technique known as Transfer Learning. Implementing a DL NN typically requires a substantial amount of data, however, collecting scratch data from vehicles with sensors can be a costly and time-consuming task, making the available data insufficient for effective testing of scratch detection algorithms. To address these challenges, various studies propose the use of fine-tuning or publicly available datasets as alternatives. By utilising CNNs such as VGG16 or ResNet50, DL can be employed with relatively small amounts of data. These NNs have been trained on massive datasets, and their weights have been made available online. Transfer Learning involves transferring the knowledge gained by these pre-trained models to different but similar problems. This is accomplished by removing the top layers responsible for classification and adopting the weights of the remaining layers as a feature extractor. These layers are typically frozen, meaning that they are not updated during training. Subsequently, the user can add a classifier for the new task and introduce additional new layers as needed. Fig. 3 illustrates a general example of transfer learning.

Fine-tuning these NNs entails refining a pre-trained model using a new dataset. During this process, images from the original training dataset are replaced by images related to the new class of interest, more specifically, spectrograms generated from the audio captured inside vehicles. Applying fine-tuning or transfer learning can be particularly useful in CNNs for the problem at hand. Therefore, refining these pre-trained NNs on large public datasets was a reasonable strategy to expedite the training process. The ImageNet dataset, on which these pre-trained NNs were trained, does not contain spectrogram-like images. Consequently, the weights might not be optimised for solving a problem that deviates from the original ImageNet challenge. Therefore, it is crucial to explore alternatives where the ImageNet dataset does not play a significant role in the developed solution.

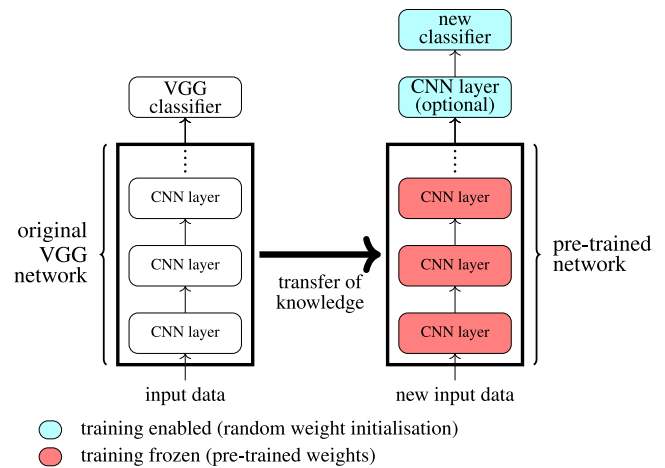


Fig. 3. A general illustration of transfer learning/knowledge. The pre-trained NN on the left is utilised as a base (without the classifier) for the NN on the right.

### 3.2. Data collection for scratch detection

The research work described in this article is focused on detecting scratching events through audio signals. Therefore, data acquisition must respect this restriction, by using a setup that contains a microphone to record the audio. The audio clips used were gathered using sensors positioned on the windshield near the rear-view mirror of various types of vehicles and under different conditions. While all scratching events were recorded with the vehicle in a stationary state, it was essential to include data from moving vehicles due to the diversity of background noise encountered in such scenarios. These dynamic situations often involve multiple passengers engaged in conversations, singing, radio listening, and similar activities. Consequently, the data collection aimed to encompass various use cases that represent the majority of sounds to which a vehicle might be exposed. These categories include:

- **Scratches:** Capturing the sounds of scratches while the vehicle is stationary.
- **Talking:** Recordings of conversations among passengers.
- **Radio sound:** Gathering audio with singing and unusual sounds that typically do not occur during normal passenger conversations.
- **Vehicle noises:** Including sounds from engines, wipers, mirrors, doors opening and closing, ventilation systems, and other vehicle-related noises.
- **Damage noises:** Incorporating abnormal events beyond scratches that could potentially be close to scratching events, such as collisions with other vehicles and impacts with objects.

This comprehensive approach to data collection ensures a diverse and representative dataset for training and testing the scratch detection algorithm. The types of events recorded in the full dataset used in this work are indicated in Tables 1–3. Table 1 lists the events that result in damages to the vehicle, but that are not scratch-related. Table 2 lists events that create sound, but that do not generate damages to the vehicle and Table 3 enumerates the scratching events that make up our target class.

Due to constraints during the data collection phase, it was not possible to capture every conceivable type of event, such as “breaking windows”. The primary focus of the collection efforts was on stationary data, as all scratch data were collected in this manner. While there were a limited number of data collections involving moving vehicles, the duration of recordings in such cases was considerably longer. This extended recording time helped compensate for the relatively lower number of collections in this category.



**Table 1**  
Damage events, excluding scratches.

Description	# events
Eoor opens against object	54
Object impacts door	37
Object impacts bumper	33
Vehicle hits bumper	5
Object impacts back trunk	1

**Table 2**  
Non-damage events.

Description	# events
Close door	532
Knock	455
Slap roof	403
Wiper is on	230
Slap windshield	201
Open/close sun visor	134
Impact object	127
Open/close makeup mirror	93
Open door	78
Mount/dismount holder	55
Open/close mirror folding	55
Slide object against windshield	42
Detach/fix sun visor	28
Open door against object	20
Stomp	6
Wash vehicle	4
Ventilation is on	3

**Table 3**  
Scratching events.

Description	# events
General scratch	106
Front	65
Back	54
Door front left	25
Door back right	24
Door front right	16
Door back left	14
Bumper front left	4

**Table 4**  
Car models used in the data collection.

Car brand	Car model	Acquisition
Audi	A5	Stationary/moving
BMW	i3	Stationary
BMW	X1	Stationary
BMW	5	Stationary
Mercedes	A45	Stationary/moving
Mercedes	GLA	Stationary
Mini		Stationary
Smart		Stationary
Volkswagen	Polo	Stationary
Volkswagen	Tiguan	Stationary

**Table 5**  
Stationary and moving data collection.

Car state	# collections	Collections (time)
Moving	9	50 min 45 s
Stationary	504	4 h 21 min 01 s

Table 4 provides details regarding the different vehicles used and the types of acquisitions. A total of 513 data collections in audio format were considered. More details can be observed in Table 5.

Since data collection was conducted using various vehicles, diverse locations, and consequently different microphones, a crucial step was to establish a common ground through post-processing. This involved

**Table 6**  
Scratching and non-scratching events distribution in data collection.

Type of event	# events	Events (time)
Non scratch	2656	40 min 02 s
Scratch	309	2 min 32 s

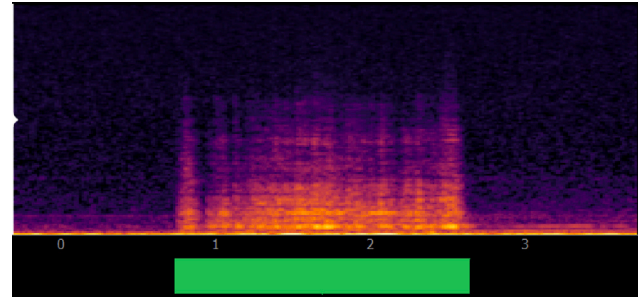


Fig. 4. Example of a label (in green).

resampling all audio clips to a consistent sampling rate of 24 kHz to ensure uniformity. Following the data collection phase, the focus shifts to examining the scratch data separately. Table 6 outlines the distribution of scratch and non-scratch data events, looking at the table one of the most notable characteristics of this dataset becomes evident: it is highly imbalanced. This mirrors a real-life scenario where the occurrence of non-scratch sounds in a vehicle significantly outweighs scratch-related events. To address this potential data imbalance, various techniques were employed.

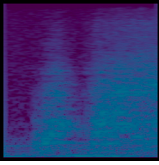
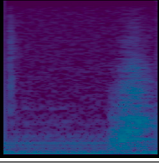
### 3.3. Labelling

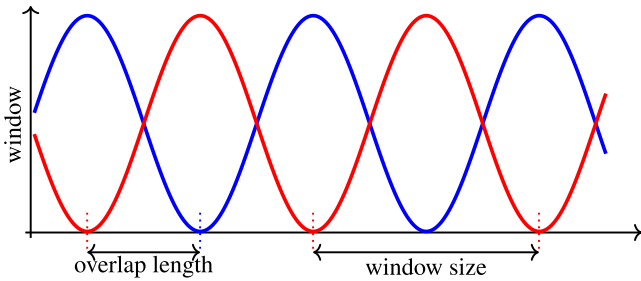
Data labelling is a time-intensive phase in the development process and arguably one of the most critical aspects in creating supervised ML algorithms as the quality of annotated data can significantly impact the effectiveness of the solution. Given its vital role in enabling the algorithm to build an accurate understanding of real-world conditions, data labelling should not be underestimated. Poorly labelled data can lead to very complex issues in the algorithm that are challenging to diagnose.

Initially, each audio clip underwent labelling with the onset and offset timestamps of scratch-related events. To ensure accurate labelling, the data labelling process was executed in two stages. The first labelling took place during the data collection phase, facilitated by a software application that allows real-time labelling using hotkeys. Subsequently, another software application provided a user-friendly interface, enabling meticulous analysis of each clip. This interface allowed for signal visualisation and playback, along with the verification of corresponding labels. Fig. 4 displays an audio clip featuring a labelled section.

After labelling the collected data clips, spectrograms were generated using soft labels that denoted the percentage of the presence of a scratching event within the respective window of analysis. For instance, consider a spectrogram corresponding to 0.5 s of audio, a data point assigned a soft label of 0.4 indicates that a scratch is present for 40% of that specific spectrogram, specifically spanning 0.2 s. This soft labelling approach offers a more accurate representation compared to hard labels, where data points are assigned to a specific class (scratch or non-scratch), the difference between both approaches can be observed in Table 7. As soft labelling is a more accurate way of representing the duration of scratches in the time interval represented by a spectrogram, this was the method used in this work.

**Table 7**  
Soft and hard labels for two different spectrograms.

Spectrogram image	Soft label	Hard label
	0.88	1
	0.28	0



**Fig. 5.** Window size and overlap length of the window function for STFT.  
Source: Adapted from Trethewey (2000).

### 3.4. Dataset specifications and time–frequency representations

Three major datasets were created to serve as training and test data. Each dataset can be viewed as an incremental step in complexity, stemming from a gradual increase in detection challenges. The first dataset addresses the problem in relatively easy conditions, with only scratching events and, therefore, zero deliberate background events other than normal background noise that the microphone would pickup during the collection phase. The second and third datasets represent a progression towards solutions capable of handling real-life scenarios with diverse background noises and weaker scratch signals. This iterative process evolved through continuous analysis and a comprehensive assessment of the insights and findings from each prior dataset iteration.

These insights drove the necessary adjustments to tackle false predictions effectively. The dataset experiences a substantial increase in size in its second version. Consequently, a custom data loader was developed to efficiently divide the data into small batches that fit into memory. This approach enables training with datasets of varying sizes while using limited physical memory. During training and validation, images are loaded in a random order to enhance generalisation in contrast to testing images that maintain their order to facilitate matching with labels during evaluation.

Several types of time–frequency images were considered in this work, all created using a sliding window technique with a window size of 500 ms and a hop length of 50% of the window size. An example of overlapping windows is shown in Fig. 5. Even though the overlapping lowers the computational performance due to more data being calculated, it is necessary and a common strategy used in the literature both in the signal processing stage and in the calculation of the short-time Fourier transform (STFT) as it minimises the effect of leakage by decreasing the signal amplitude near the boundaries of the samples (Trethewey, 2000).

The conversion process from audio to image represents one of the most critical aspects of this work. Given the half-second window analysis of audio signals, it was imperative to select the right features

**Table 8**  
Layout of all datasets considered.

Dataset	Training	Validation	Testing
#1	913	323	533
#2	36 927	12 591	22 185
#3	37 612	12 686	22 318

**Table 9**  
Detailed description of the datasets.

	Dataset #1	Dataset #2	Dataset #3
Data collection files	91	465	513
Scratch-related events	166	249	309
Non-scratch-related events	0	2 635	2 656
Scratch-related images	361	553	714
Non-scratch-related images	1408	71 150	71 902

for NN input from the previously studied transformations. In this study, possible signal transformations include gammatoneagrams, log spectrograms, log-mel spectrograms, and chromagrams. Fig. 6 illustrates an example of all these four transformations for the same scratching event. In each case, various parameters were thoroughly explored. For example, different values for the STFT in creating log and log-mel spectrograms can significantly impact the accuracy of the NNs. These parameters play a pivotal role in determining the size and characteristics of the spectrograms.

In the process of generating spectrogram images for the dataset, a crucial step involved the careful normalisation of the data. Prior to image creation, the maximum and minimum values within each matrix representing the spectrograms were computed across the entire dataset. This ensures that the data fall within a standardised range promoting stability and efficiency in the training process and contributing to improved model performance and generalisation.

All data were separated into training, validation and testing with each group getting, approximately, 50%, 20% and 30% of the images. The final layout of the dataset is shown in Table 8.

#### 3.4.1. Dataset #1

The first dataset serves as the initial checkpoint in our data collection efforts, aiming to replicate relatively simple scenarios where scratches are highly distinguishable with minimal or no background noise, some examples of that kind of scratches are illustrated in Fig. 7. While not representative of real-life situations, this dataset plays a crucial role in gradually developing the algorithm by gaining insights into data characteristics and detection capabilities. This dataset comprises 91 distinct recorded clips from our data collection phase, resulting in 166 labelled scratching events with no other types of events or damages annotated. Within this dataset, 1769 log-mel spectrogram images are generated. A more comprehensive overview of this dataset is provided in Table 9.

This dataset marks the starting point for increasing the complexity of scratch detection. Its smaller image count, as shown in the distribution presented in Table 10, was important in facilitating faster training speeds, allowing for more extensive experimentation with high-level changes to the training data. This process aids in identifying major dataset flaws, optimal augmentations, and the most effective time–frequency feature within options like the mel-spectrogram, spectrogram, chromatogram, and gammatoneagram. While the dataset ideally separates images into either clear scratch data or low-noise background, it may still include some false positives due to imperfect data collection, including occasional talking and other noises that introduce challenging-to-predict data. These experiments essentially serve as a proof of concept, informing the progression to dataset 2.

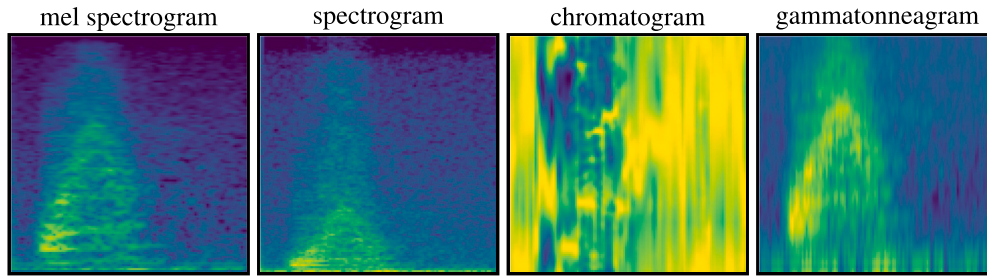


Fig. 6. Different types of visual representations considered for the same scratch data.

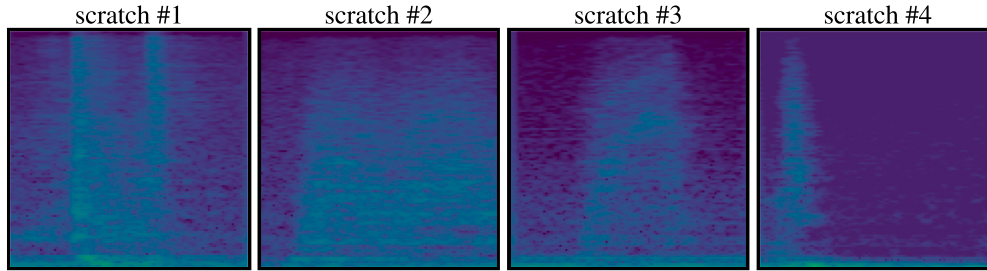


Fig. 7. Example of scratches being pictured in log-mel spectrograms.

Table 10

Distribution of images in training/validation/testing sets featured in dataset 1.

Dataset	Scratch	Background
Training	208	705
Validation	60	263
Testing	93	440

Table 11

Distribution of images in training/validation/testing sets featured in dataset 2.

Dataset	Scratch	Background
Training	339	36 588
Validation	90	12 501
Testing	124	22 061

### 3.4.2. Dataset #2

The second checkpoint signifies a transition to a more diverse dataset. The initial dataset revealed challenges in handling random background noise, prompting this iteration to incorporate background data in an effort to reduce the occurrence of false positives. This dataset introduces a wide variety of non-scratch damage events and includes examples of common noises originating from vehicle engines, passenger conversations, and live radio music, some examples of that kind of background events are illustrated in Fig. 8. A detailed breakdown of the dataset is provided in Table 9.

This dataset was specifically designed to address the issue of false positives detected in dataset #1. It encompasses different types of non-scratch-related events, with careful consideration given to ensuring that each type of background event is represented in both the training, validation and testing datasets. However, the substantial increase in the number of background images contributes to a highly unbalanced dataset, as illustrated in Table 11.

### 3.4.3. Dataset #3

The third and final dataset in this work aims to replicate real-life conditions as closely as possible. As preliminary tests showed some difficulty in dealing with distant background talking, the need to search and add this type of data arose for a better and more complete final dataset version. While it maintains a similar number of total images

Table 12

Distribution of images in training/validation/testing sets featured in dataset 3.

Dataset	Scratch	Background
Training	438	37 174
Validation	109	12 577
Testing	167	22 151

as dataset #2, this version introduces a more diverse range of scratch-related events, including those with weaker audio signals, resulting in dimmer representations of scratching events within the spectrograms. Refer to Fig. 9 for examples of these scratches. The inclusion of such variations is essential to challenge the model and ensure its robustness in detecting scratches in realistic scenarios. A more detailed description of the dataset can be seen in Table 12.

### 3.5. Metrics

As the development of the algorithm progresses, it becomes increasingly important to establish robust metrics for evaluating and comparing its performance. The issue of classifying images for damage detection purposes can be framed as a binary classification task, where the goal is to distinguish between damage and non-damage instances. This binary classification leads to four possible outcomes: true negative (TN), true positive (TP), false negative (FN), and false positive (FP). These outcomes can be expressed by a two-by-two matrix known as a confusion matrix (Fig. 10).

While various metrics can be derived from a confusion matrix, some of them can be misleading, especially when dealing with extremely unbalanced datasets, as is the case in this work, where the number of background event samples significantly outweighs the number of scratch data samples. To address this challenge, the MCC was selected as the primary metric for evaluating the CNN models, as it is considered the most robust and recommended metric for binary classification performance evaluation (Canbek, Temizel, & Sagioglu, 2021). The MCC formula looks like this:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (1)$$

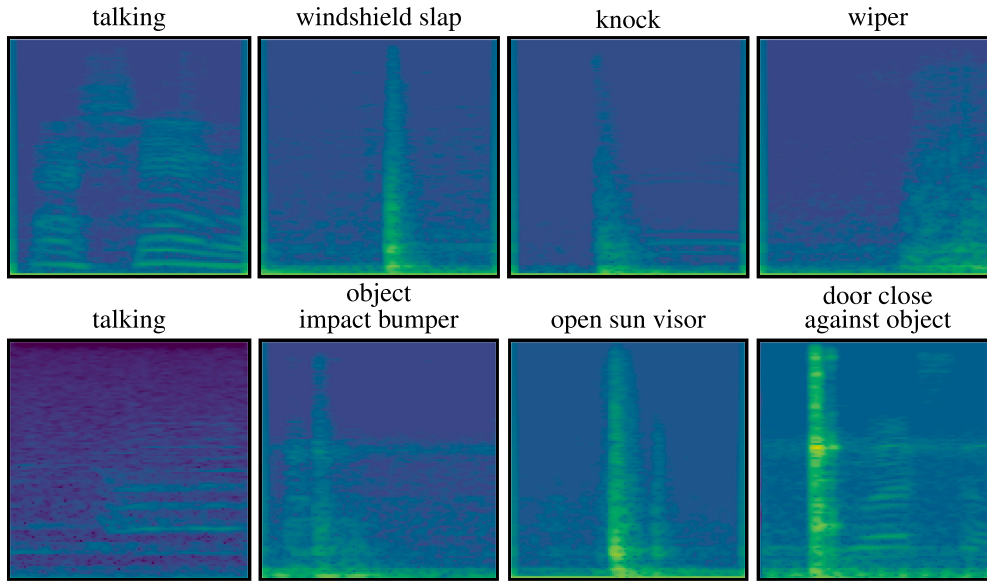


Fig. 8. Example of different types of backgrounds events being pictured in log-mel spectrograms.

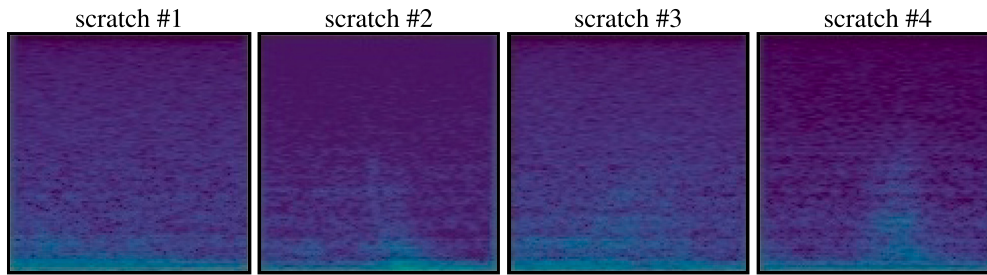


Fig. 9. Example of more complex scratches being pictured in log-mel spectrograms.

		true class	
		positive	negative
predicted class	positive	TP	FP
	negative	FN	TN

Fig. 10. Confusion matrix for binary classification.

The MCC scores range between  $-1$  and  $+1$ , in which those extreme values correspond to misclassification and perfect classification, respectively. The 0 value indicates a random prediction, which can be replaced by a coin tossing classifier. MCC is a robust statistical measure that offers several advantages over metrics like F1 or accuracy (Chicco & Jurman, 2020). MCC yields a high score only when the model performs well across all four categories of the confusion matrix, making it proportional to the sizes of both positive and negative elements in the dataset. This characteristic makes MCC particularly suitable for assessing model performance in unbalanced datasets.

### 3.5.1. Establishing a threshold

In the context of developing a predictive model, class prediction is achieved through the use of a threshold, often set at 0.5. Values equal to or greater than the threshold are classified as scratches, while values

below the threshold are classified as non-damage instances. However, it is crucial to recognise that using a default threshold of 0.5 may not be optimal for every problem, thus, the fact that this default threshold is commonly used in the literature it is important to do some threshold tuning and analysis.

To optimise the models, it was conducted a threshold analysis. Fig. 11 illustrates the distribution of scratch-related images based on their soft labels. The distribution appears linear until reaching a threshold value of 0.4, beyond which there is a more exponential increase. Consequently, we selected 0.4 as the threshold that distinguishes scratch predictions from background/non-scratch predictions. Any image with a label equal to or greater than 0.4 is considered a scratch, while those with labels below this value are categorised as background.

This threshold analysis was conducted using the full dataset, as intended in dataset 3, and remained constant throughout the entire development process. Additionally, similar inferences were made when conducting the same analysis on dataset 1, even though it contained a smaller number of scratches. This approach ensured a consistent and data-driven threshold selection for evaluating the performance of our models across different datasets and conditions.

### 3.5.2. Image-oriented metric

An image-oriented metric adheres to the standard evaluation process for image classification in DL models. Each real label of an image is compared to its predicted label, and based on a predefined threshold, it is categorised into one of the four confusion matrix categories. This metric offers a realistic assessment of the NN effectiveness, as it scrutinises each image individually. This approach stands in contrast to the scratch-oriented metric, which is discussed subsequently.



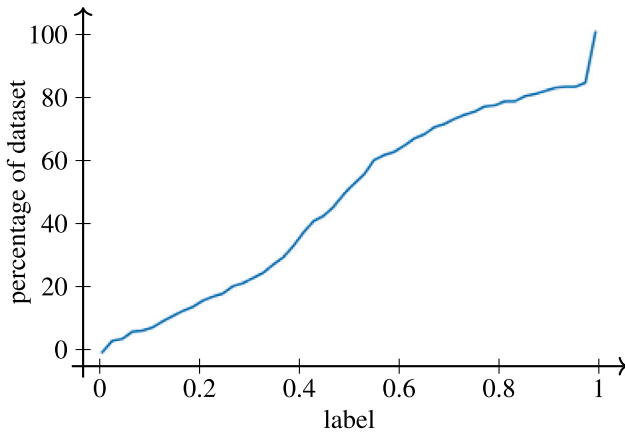


Fig. 11. Accumulative percentage of scratches by label (only for images with label higher than 0).

### 3.5.3. Scratch-oriented metric

The scratch-oriented metric introduces a novel perspective on the classification of positives within the algorithm. Given that the audio stream is analysed in a sliding window fashion with a fixed length, a single scratching event may generate multiple spectrograms. In an image-oriented metric, a true positive in the confusion matrix indicates an image for which both the true and predicted labels are higher than the designated threshold, classifying it as a scratch. While this image-oriented classification is valid and informative, it is crucial to evaluate the performance of the algorithm in real-life scenarios, in which the primary goal is to detect scratches. In practical terms, if a scratch consists of multiple images, it might not be critical if one or two of these images fall below the detection threshold. What truly matters is that the algorithm successfully identifies the presence of a scratch within the entire set of images comprising that particular scratching event.

Consequently, we developed a scratch-oriented classification, where a true positive signifies a correct prediction of a scratching event as a whole, as opposed to a correct prediction for individual images within the scratch. This approach aligns more closely with the end goal of detecting scratches comprehensively. For instance, if a scratching event is composed of four different images, it is considered a true positive if the algorithm successfully identifies the presence of the scratch in any of these images. This approach provides a more holistic evaluation of the algorithm, ensuring that scratches are identified across all relevant images within a given event.

### 3.6. Exploring spectrogram mixing

In an attempt to merge features of multiple types of visual representations, it was developed a technique where each generated image would be a fusion of three different images. As the images being created are in RGB format, and RGB images are essentially 3D matrices with three channels for red, green, and blue colours, the project took advantage of this structure to incorporate three distinct matrices of different features, placing each one in a separate channel. Although this might result in a spectrogram that appears imperceptible to the human eye, the expectation was that the CNN would learn the fundamental changes between scratches and background data presented in the input data matrices. In the first iteration of this spectrogram mixing approach, a non-mel-scaled spectrogram, a mel-scaled spectrogram, and a gammatoneagram were combined. An example of this process is presented in Fig. 12.

The chromagram was introduced in the second iteration and integrated with a log-mel-scaled spectrogram and a gammatoneagram. This approach aimed to further enrich the input data and enhance the NN

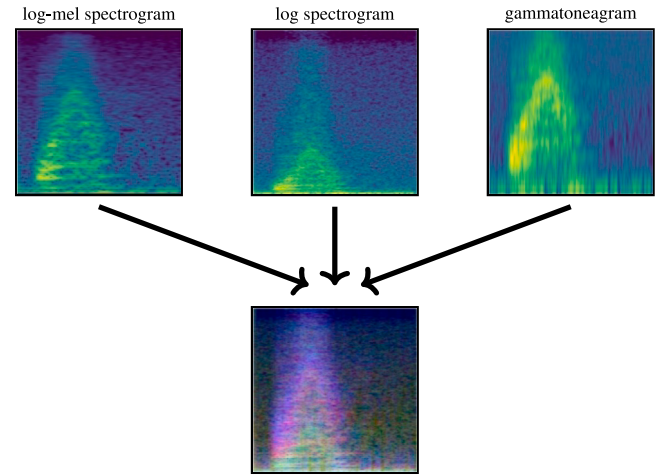


Fig. 12. Example of log spectrogram, log-mel spectrogram and gammatoneagram mixing.

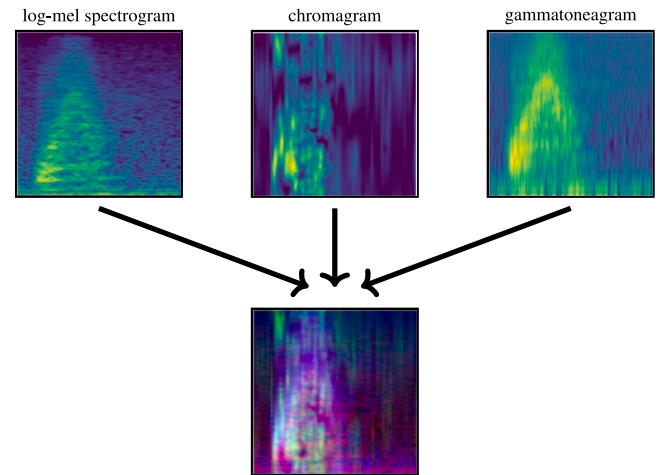


Fig. 13. Example of log-mel spectrogram, chromagram, and gammatoneagram mixing.

ability to extract relevant features. An example of the mixing process in this second iteration is depicted in Fig. 13.

Spectrogram mixing represents an innovative approach to data representation, where diverse spectrogram types are integrated into a single image, potentially providing the CNN with a more comprehensive understanding of the intricate features of the audio data, ultimately contributing to improved scratch detection accuracy.

### 3.7. Exploring training set augmentations

When initially reached a validation loss plateau during training, it became imperative to consider a variety of data augmentation techniques. Given the relatively limited amount of scratch data available, data augmentation emerged as a crucial strategy to increase the diversity of the training dataset by applying random yet realistic transformations. To achieve this, it was closely examined the spectrogram images generated from the existing datasets and rigorously tested different relevant augmentation methods on them. These methods were evaluated individually for their effectiveness before combining them to explore whether more promising results could be achieved. All augmentations were carried out using the *imgaug* library.

- **Flip augmentation:** Our initial experiments focused on image flips. Horizontal flips, in particular, were applied, which depict

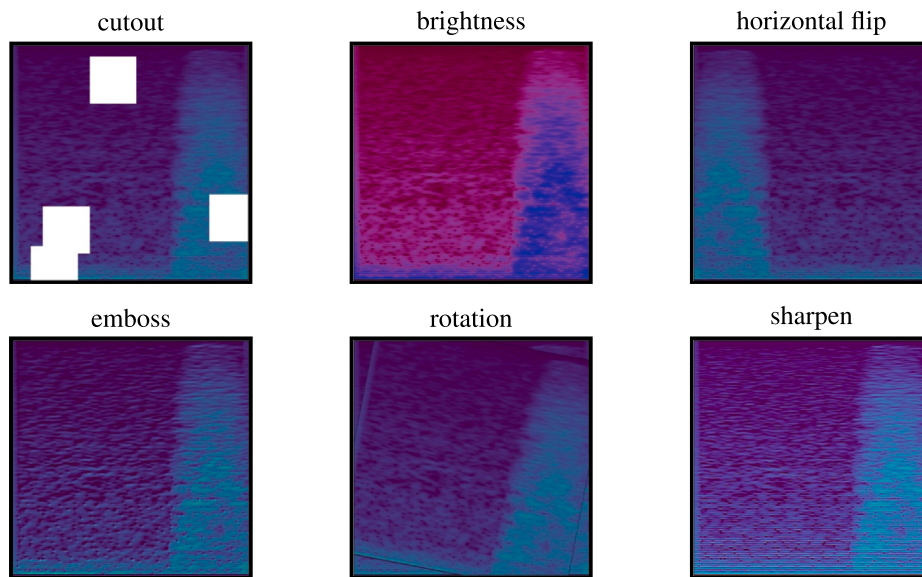


Fig. 14. Multiple type of augmentations used in this work.

a scratch sound in reverse while preserving all the essential characteristics of the spectrogram. Remarkably, flip augmentation consistently proved to be the most effective technique in enhancing scratch detection throughout each iteration of our datasets.

- **Emboss augmentation:** The next augmentation involved embossing the spectrogram image. This technique accentuates and highlights pixels in areas with high contrasts, as if the image were etched onto a metal plate. Embossing is similar to sharpening as it aims to emphasise scratch-related regions while diminishing background static data. However, when employed as a standalone augmentation, it did not yield significant improvements compared to flip augmentations.
- **Sharpen augmentation:** Another augmentation method we explored was image sharpening. Similar to embossing, sharpening sought to enhance scratch features. Although there was a discernible improvement in results compared to the baseline system, these enhancements did not match the effectiveness of flip augmentations.
- **Rotation augmentation:** We also experimented with random rotations as an augmentation technique. This involved randomly rotating an image clockwise or anti-clockwise within a range of 0 to 30° degrees, causing the spectrogram position inside the frame to change.
- **Brightness augmentation:** Another augmentation strategy we investigated was adjusting the brightness of the image. Detecting subtle scratches poses one of the primary challenges in this research. Manipulating image brightness aimed to create new spectrograms where features were either more or less pronounced compared to those under normal lighting conditions.
- **Cutout augmentation:** Cutout augmentation involves blocking out rectangular regions of the spectrogram image. This technique was tested to assess its impact on improving the model performance (Devries & Taylor, 2017).

Fig. 14 shows examples of these augmentations within the same spectrogram.

#### 4. Selecting the CNN and its parameters

Selecting an appropriate CNN architecture is a pivotal step in the development of the detection algorithm. Choosing the wrong NN can significantly impact results and potentially lead to a slow, tiresome

and frustrating development process. Therefore, it is imperative to deconstruct the problem, to analyse it while considering available resources, and to weigh key properties before making a decision:

- **Input and type of problem:** Since the objective is to compute spectrograms and feed them to ANNs in the form of images, it is logical to explore solutions that have proven effective for computer vision problems, such as CNNs.
- **Computational power:** Another crucial factor influencing the choice is the computational capacity of the deployment environment. While a high-performance algorithm can yield excellent results, a heavyweight and sluggish architecture could pose challenges during real-life implementation on devices with limited computational resources.
- **Inference time:** A primary objective of this work is to implement the CNN algorithm in a real-life scenario, where scratch detection occurs in real-time. Given that the audio analysis is conducted in small half-second portions, this highlights the importance of a fast inference speed.

Having established that mel spectrograms in the form of images are to be used for an image classification task, we turn to state-of-the-art research to identify potential CNN architectures that excel in image classification with transfer learning.

All the development was conducted with the **VGG16** architecture. VGG16 includes 16 layers and has a considerable size with 130 million parameters, which makes it less attractive. However, it is also a NN widely studied in the literature for computer vision tasks. The final architecture of the VGG16 model used in this work is depicted in Fig. 15.

Beyond the choice of NNs for experimentation, several fundamental principles and techniques related to DL deserve mention and exploration:

- **Activation function:** The activation function determines whether a neuron should be activated or not. It introduces non-linearity to the model, enabling it to learn complex mappings from inputs to outputs. Common activation functions include ReLU (rectified linear activation), sigmoid, and tanh. For this project, it was opted the sigmoid activation function because it effectively maps the NN output to a bounded range between 0 and 1. Given that the task involves binary classification (detecting scratches or not) with soft labels between 0 and 1 as well, the sigmoid activation function is

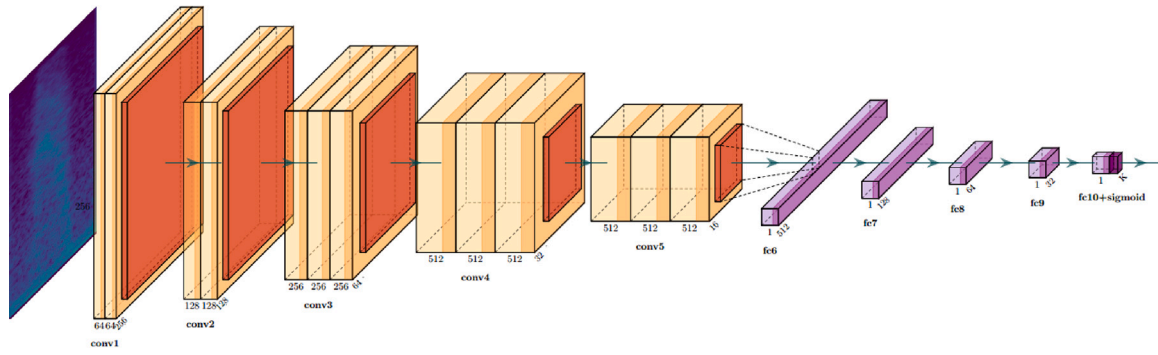


Fig. 15. The architecture of the VGG16 model.

well-suited for producing probability-like values that indicate the likelihood of a scratch.

- **Loss function:** The loss function calculates the error between the predicted and actual values. It quantifies how well the model is performing. In classification tasks, common loss functions include categorical cross-entropy and binary cross-entropy. These functions measure the dissimilarity between predicted and actual class probabilities. For this work, the log-cosh loss function is employed.

## 5. Results

This section discusses the three major iterations that were followed to benchmark and analyse the approach described in the article, according to the metrics established to evaluate the respective performance. Each iteration uses one of the datasets presented previously.

### 5.1. iteration #1

The first iteration used dataset #1 and included a series of comprehensive experiments, laying the foundation for subsequent iterations. To identify the optimal solution for the proposed problem, we initially focused on creating a simpler environment that allowed us to explore and effectively find the best useable features. These experiments primarily centred on testing different types of spectrograms and Fourier transforms, which shape these spectrograms. Throughout these experiments, we employed the VGG16 model, leveraging its default weights from the ImageNet dataset as a feature extractor. The extracted features were then passed through a final decision layer, applying a sigmoid function to constrain the output within the interval [0,1] for binary classification: background or scratch, using the threshold established in Section 3.5.1.

#### 5.1.1. Time-frequency representations

The initial investigation revolved around determining the most suitable image representations for feeding the CNN. Following the consensus in state-of-the-art research, which consistently advocated for log-mel spectrograms in sound event detection problems (Atsavasilert et al., 2019; Nguyen, Lin, & Huang, 2023), we focused on these representations. This experiment involved computing various types of spectrograms using a sliding window of 0.5 s in length with a 50% overlap. It was also at this stage that we conducted the experiments described in Section 3.6, in which we used a combination of various time-frequency representations in a single data point, as shown in Table 13.

#### 5.1.2. Fourier transform

Building upon the discovery that log-mel spectrograms yielded superior results, we proceeded to explore different transforms within the same length of analysis with this representation. Leveraging the capabilities of the librosa library, we fine-tuned different parameters for computing mel-scaled spectrograms while maintaining the window of analysis. The summarised results are presented in Table 14.

Table 13

Results for different types of spectrograms.

Representation	Image MCC	Scratch MCC
Log	0.85	0.79
Log-mel	<b>0.90</b>	<b>0.90</b>
Gamma	0.87	0.88
Chroma	0.72	0.73
Log-mel+mel+gamma	0.85	0.83
Log-mel+chroma+gamma	0.88	0.85

Table 14

Results for different types of mel-scaled spectrograms.

Fourier transform	Image MCC	Scratch MCC
256 × 256	<b>0.90</b>	<b>0.90</b>
224 × 224	0.83	0.82
60 × 360	0.77	0.84

Table 15

Results for different types of mel-scaled spectrograms.

Augmentation	Image MCC	Scratch MCC
No augmentation	0.90	<b>0.90</b>
Horizontal flip	<b>0.92</b>	0.88
Rotation	0.84	0.87
Cutout	0.90	0.88
Emboss	0.83	0.86
Sharpening	0.86	0.88
Whitening	0.82	0.84

#### 5.1.3. Data augmentation

To evaluate the impact of data augmentation, a set of augmentation techniques was applied during training. Table 15 summarises the outcomes, shedding light on the impact of the augmentations presented in Section 3.7.

### 5.2. iteration #2

Building upon the knowledge gained from the results in iteration #1, iteration #2 aimed to further refine our algorithm performance in more complex and realistic scenarios. This second iteration was based on dataset #2 and emphasised the inclusion of various potential background events that could confound the NN in detecting scratches. This was the part where most of development time was spent. Additionally, iteration #2 introduced a substantial increase in the volume of data, with over 70 000 images, of which only 553 were associated with scratches. Due to the lessons learned in iteration #1, we exclusively used log-mel spectrograms from this point on.

**Table 16**

Results for different augmentations and batch sizes in iteration #2.

Batch size	No augmentation		Horizontal flip		Cutout		All augmentations	
	S_MCC	I_MCC	S_MCC	I_MCC	S_MCC	I_MCC	S_MCC	I_MCC
8	0.84	0.87	0.85	0.89	0.76	0.85	0.87	0.88
16	0.81	0.86	0.84	<b>0.90</b>	0.87	0.88	0.85	0.88
32	0.80	0.86	0.86	<b>0.90</b>	0.77	0.83	0.84	0.85
64	0.87	0.88	0.80	0.87	0.88	0.87	<b>0.90</b>	0.85
128	0.85	0.88	0.84	0.88	0.84	0.86	0.85	0.85

**Table 17**

Results for different augmentations and batch sizes with no pre-trained weights.

Network specifications	S_MCC	I_MCC
Batch size 32 - horizontal flip only	<b>0.92</b>	<b>0.90</b>
Batch size 64 - all augmentations	0.90	0.89

### 5.2.1. Data augmentation and batch size

In this phase, we carried forward the augmentation techniques that yielded the best results in iteration #1: cutout and horizontal flips. We also introduced a new augmentation that combined all of the augmentations previously tried. To explore the impact of batch size on model performance, we conducted experiments with batch sizes ranging from 8 to 128. The summarised results can be found in Table 16.

### 5.2.2. Base model fully trainable

Until this point, our model training process had been initiated with pre-trained weights sourced from the ImageNet dataset. While this is a widely accepted practice that helps accelerate the training process, it was important to assess whether these pre-trained weights were optimally suited for our spectrogram-based problem. Given the limitations of the ImageNet dataset, which does not offer a rich variety of spectrogram-like images, we conducted an experiment where we initialised the model without pre-trained weights. This experiment aimed to determine if pre-trained weights might not align with our problem performance requirements, particularly in relation to spectrograms. Once we had already looked for what worked better in terms of augmentations and batch sizes, this experiment was done on the best results observed in Table 16. The results of this experiment are exhibited in Table 17.

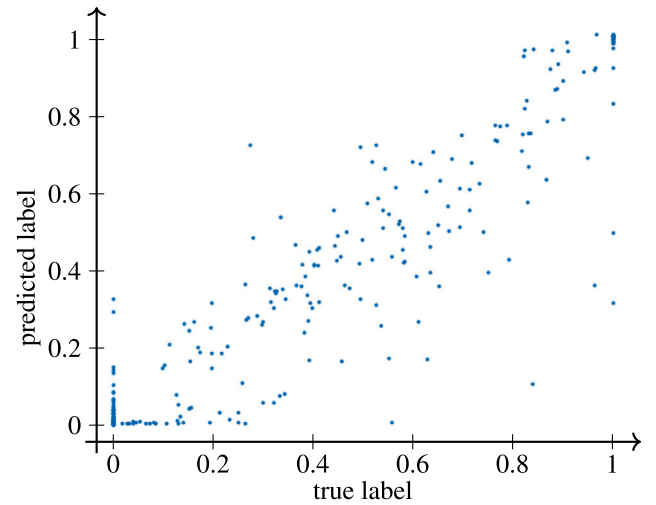
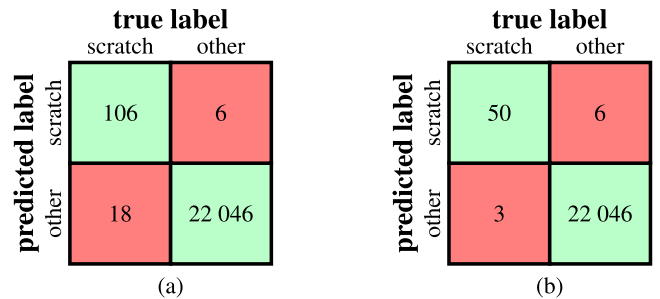
### 5.2.3. Best model results

Fig. 16 illustrates the distribution of true and predicted labels on the best model, i.e., horizontal flip augmentation with size 32 batch size in a fully trainable model.

The graphical representation plots the predicted labels on the x-axis against the true labels on the y-axis, offering valuable insights into the model performance. In an ideal scenario, where the predictions perfectly match the true labels, the graphic would exhibit a straight line of dots following the equation  $y = x$ . Deviations from this line indicate discrepancies between the predictions and the true labels, shedding light on the model accuracy. Analysing these deviations were fundamental in guiding decisions for the type of data that should be added next. A possible approach for this analysis is individually looking at false negatives and positives of the confusion matrices. Fig. 17 provides a visual representation of the classification performance for iteration #2, using both image-oriented and scratch-oriented metrics.

### 5.3. iteration #3

This iteration used dataset #3, which slightly expands the overall knowledge of different types of scratches. This is an attempt of generalising the NN into recognising harder-to-detect soft scratches. In light of the escalating training time, we decided to carry forward the best-performing elements from previous dataset iterations. Consequently,

**Fig. 16.** Predicted labels vs. true labels for the best model in iteration #2.**Fig. 17.** Confusion matrix for (a) image-oriented and (b) scratch-oriented metrics for the best model in iteration #2.

we experimented with batch sizes of 32 and 64, considering both fully trained and pre-trained models, as the ideal approach was not evident from prior experiments.

## 6. Discussion

In the realm of potential options explored within iteration #1, it became evident that log-mel spectrograms were the optimal choice. The decision to use squared images with a resolution of  $256 \times 256$ , more closely mirroring the dimensions of the ImageNet dataset, produced superior results. This departure from conventional spectrograms, which are often lengthier in the x-axis and shorter in the y-axis, proved to be a noteworthy improvement. Among the augmentation techniques tested, horizontal flip emerged as the most effective with cutout also showing promising results. In general, augmentations did not help in improving scratch detection, with the majority of the augmentations showing worse results compared to when no augmentation was applied.

The focus was always on reducing false positives. The examination of false positives in iteration #1 revealed that numerous misclassified spectrograms were shared by all trained NNs. Upon closer inspection of the corresponding audio data, it was discovered that many of these spectrograms captured events that were not initially labelled but were inadvertently recorded during the data collection process. These unanticipated events included instances of people talking during the data collection process, as well as various background noises such as objects dropping on the ground.

However, it is important to emphasise that this iteration primarily served as a foundational step in the development process. It allowed us to fine-tune crucial parameters, assess different spectrogram types, and refine data augmentation techniques. As we transitioned to iteration



**Table 18**

Results for different augmentations and batch sizes on dataset 3.

Batch size	Pre-trained				Fully-trained			
	Horizontal flip		All augmentations		Horizontal flip		All augmentations	
	S_MCC	I_MCC	S_MCC	I_MCC	S_MCC	I_MCC	S_MCC	I_MCC
32	0.74	0.82	0.85	0.83	0.79	0.83	<b>0.90</b>	<b>0.88</b>
64	0.75	0.83	0.77	0.82	0.84	0.82	0.82	0.81

#2, we built upon the insights gained here to tackle more complex and real-life scenarios, therefore recognising the significance of addressing the background sounds and unaccounted for events that had inadvertently impacted the training process in iteration #1. To mitigate these issues, deliberate efforts were made to include data segments that encompassed various background noises such as described in Section 3.4.2 and as detailed in the following sections.

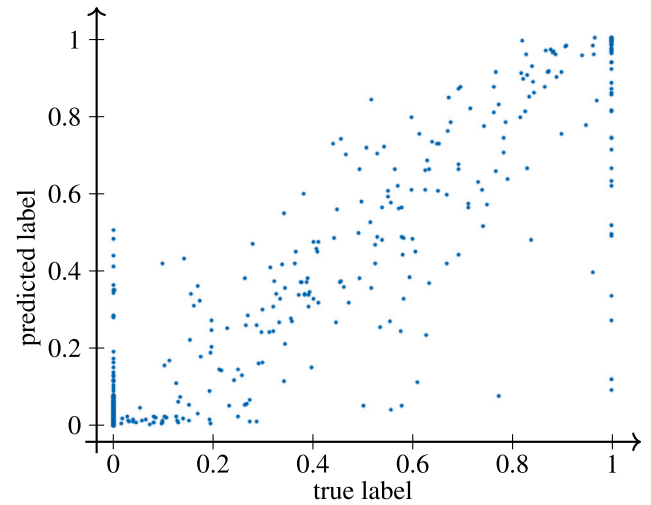
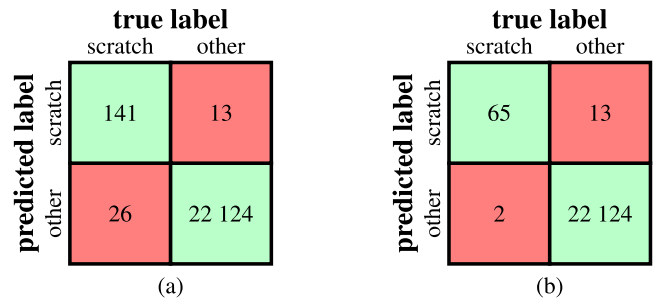
The results obtained from iteration #2 indicate that batch sizes of 32 and 64 yielded the most successful outcomes when combined with either horizontal flips or all augmentations applied simultaneously. Notably, it was found that training the base model from scratch without the pre-trained weights yielded the best results with both NNs showing improvements in both the scratch- and image-oriented metric compared to their pre-trained counterparts. All these configurations that demonstrate superior scratch detection performance were considered as the work proceeded to iteration #3. While iteration #2 witnessed a substantial expansion in terms of dataset size compared to iteration #1, the primary objective of mitigating false positives was successfully accomplished. Our attention shifted towards the analysis of false negatives. These false negatives predominantly consisted of low-amplitude and subtle sound scratches. To address this limitation and further refine the sensitivity of the model, such challenging cases were prioritised for inclusion in iteration #3.

Iteration #2 allowed us to refine our approach by incorporating a more diverse set of background events and augmentations, bringing us one step closer to the development of a robust algorithm for real-life scenarios. The lessons learned here were instrumental in shaping the next phase of the research.

The results from iteration #3, designed to emulate real-world conditions closely, underscore several important findings. Examining the outcomes presented in Table 18, it is evident that the combination of all augmentation techniques detailed in Section 3.7 consistently produced the most favourable results in both the image-oriented and scratch-oriented metrics. Unfortunately, the performance in iteration #3 did not meet the same standards as iteration #2, indicating that the NN faced challenges in effectively detecting new low-amplitude scratches in this more complex real-world scenario. We can see in Fig. 18 that the distribution of predicted vs. true labels for the best model in iteration #3 is worse than the equivalent iteration #2 represented in Fig. 16. This goes in line with the increase of false positives and overall the number of false negatives in an individual image analysis only for a small decrease in false negatives in a scratch-oriented analysis. These results for the best model are displayed, as is usual, in the confusion matrices in Fig. 19.

## 7. Conclusions

This article describes the development of a system that detects scratches in vehicles. While many such systems focus on more commonplace sounds, such as music or animal derived audio, scratch detection proved itself a new distinctive challenge, necessitating a phased approach in using the provided dataset. Our initial emphasis focused on studying DL techniques and also looking at possible transformations of raw audio data into visual representations suitable to accurate classification and detection. This led to the conclusion that the use of CNNs would be the main solution to explore for the problem at hand.

**Fig. 18.** Predicted labels vs. true labels for the best model in iteration #3.**Fig. 19.** Confusion matrix for (a) image-oriented and (b) scratch-oriented metrics for the best model in iteration #3.

Throughout the developmental stages, log-mel spectrograms stood out as the optimal choice for scratching events detection within the CNN framework. Furthermore, our in-depth exploration of NN architectures shed light that pre-trained ones, while widely acclaimed in the realm of DL, did not consistently prove as advantageous as anticipated. Indeed, the best results materialised when the NNs were left fully trainable, allowing them to adapt more effectively to the complexities of the scratch sounds.

With respect to the two objectives established for this work, the following conclusions can be indicated:

- O1: It was proven that **scratch detection within the context of vehicular audio is possible** and that CNNs can distinguish the acoustic signatures of scratches by analysing images. By splitting the provided dataset into three segments, a foundational understanding of the problem space was achieved with the first experiments. Dataset 1 validated the feasibility of scratch detection and facilitated the exploration of various data augmentation techniques and spectrograms.
- O2: It was possible to **develop an algorithm for scratch detection in vehicles that can handle more realistic scenarios**. Building upon the success obtained in iteration #1, the next two iterations (#2 and #3) addressed the problem in more diverse environments, confirming the potential to achieve positive outcomes in scratch detection, even in the presence of numerous concurrent sound events. This accomplishment reaffirms the viability of employing contemporary DL approaches for complex sound classification and detections tasks. The CNN-based algorithm achieved MCC scores of over 0.90 for the best models. This is considered an excellent value, as it is relatively close to the

highest possible value (+1.0), which means that the developed algorithm is well-suited for many industrial applications.

While successful in addressing both objectives (O1 and O2), the work faced challenges in improving the final MCC results when using the full dataset. This limitation, primarily attributed to the difficulty NNs encounter in accurately detecting scratches characterised by low amplitude and weak signals, highlights the impact of a very low signal-to-noise ratio.

While the developed CNN can be seen as promising, there is still work to be done. The field of audio-based vehicle scratch detection presents numerous opportunities for improvement. The unbalanced nature of the datasets, with a significantly higher number of background events, remains a hurdle to overcome. In future work, techniques, such as different types of data augmentation, sampling strategies, and active learning, could be explored to address this imbalance and enhance the performance.

In hindsight, it is evident that the efforts made to train the models to handle scratches with low signal to noise ratios were not adequately supported by a sufficient quantity of relevant data. This deficiency in representative examples limited the ability of the NN to successfully learn to identify them. The datasets should continue evolving to simulate real-world scenarios more accurately. This involves the inclusion of an even broader range of background events and more scratch signals.

To reach higher values for the MCC metric and thus improve the relevance and accuracy of the CNN algorithm, it is imperative to consider comprehensive data collection, emphasising the inclusion of diverse samples that better reflect these subtle scratch types. Moreover, the continuous refinement of model architectures and the fine-tuning of hyper-parameters hold the potential to contribute significantly to bridging the gap towards the desired MCC threshold, despite the relatively limited emphasis on these aspects during the developmental stages.

Considering the ever-evolving landscape of DL, exploring new model architectures holds significant promise. Consideration of more advanced model architectures, such as Resnet (He, Zhang, Ren, & Sun, 2016) and EfficientNet (Tan & Le, 2019), is promising. These models, known for their strong performance in image classification tasks, could provide advantages in accuracy and computational efficiency. While conventional CNNs have been instrumental so far, the integration of transformer-based models, renowned for their proficiency in sequential data tasks, represents an intriguing alternative. The adoption of models like the Vision Transformer (ViT) or its successors can potentially bring improvements in vehicle scratch detection considering their promise at capturing intricate patterns in data.

In summary, the use of the CNN presented in this study represents a significant step towards developing an automated system for accurate identification of scratching events. With further advancements and integration of complementary methods, the CNN-based algorithm can serve as a valuable and feasible solution for car rental companies, car manufacturers, and other players in the automotive domain.

#### CRedit authorship contribution statement

**André R. Soares:** Performed conceptualisation, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Visualisation. **André L. Ferreira:** Performed conceptualisation, Methodology, Formal analysis, investigation, Resources, Data curation, Writing – review & editing. **João M. Fernandes:** Responsible for writing – original draft, Writing – review & editing, Visualisation.

#### Declaration of competing interest

The authors declare that they have no conflicts of interest or personal relationships that could have appeared to influence the work reported in this paper.

#### Acknowledgements

This work has been supported by FCT—Fundação para a Ciência e a Tecnologia within the R&D Units Project Scope UIDB/00319/2020.

#### Data availability

The data that support the findings of this study are available from Bosch Car Multimedia Portugal S.A. Still, restrictions apply to the availability of these data, which are not publicly available. Data are, however, available from the authors upon reasonable request and with permission of Bosch Car Multimedia Portugal.

#### References

- Aloysius, N., & Geetha, M. (2017). A review on deep convolutional neural networks. In *6th international conference on communication and signal processing* (pp. 588–592). IEEE, <http://dx.doi.org/10.1109/ICCSP.2017.8286426>.
- Arnal, L., Solanes, J. E., Molina, J., & Tornero, J. (2017). Detecting dings and dents on specular car body surfaces based on optical flow. *Journal of Manufacturing Systems*, 45, 306–321. <http://dx.doi.org/10.1016/j.jmsy.2017.07.006>.
- Atsavasirile, K., Theeramunkong, T., Usanavasin, S., Rugchatjaroen, A., Boonkla, S., Karnjana, J., et al. (2019). A light-weight deep convolutional neural network for speech emotion recognition using mel-spectrograms. In *14th international joint symposium on artificial intelligence and natural language processing* (pp. 1–4). <http://dx.doi.org/10.1109/ISA-NLP48611.2019.9045511>.
- Canbek, G., Temizel, T. T., & Sagirolu, S. (2021). Benchmetrics: A systematic benchmarking method for binary classification performance metrics. *Neural Computing and Applications*, 33, 14623–14650. <http://dx.doi.org/10.1007/s00521-022-08049-9>.
- Chico, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, 21, <http://dx.doi.org/10.1186/s12864-019-6413-7>.
- Coelho, G., Matos, L. M., Pereira, P. J., Ferreira, A., Pilastrri, A., & Cortez, P. (2022). Deep autoencoders for acoustic anomaly detection: Experiments with working machine and in-vehicle audio. *Neural Computing and Applications*, 34, 19485–19499. <http://dx.doi.org/10.1007/s00521-022-07375-2>.
- Devries, T., & Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. <http://dx.doi.org/10.48550/arXiv.1708.04552>, CoRR abs/1708.04552.
- Din, N. M. U., Assad, A., Dar, R. A., Rasool, M., Sabha, S. U., Majeed, T., et al. (2024). Ricenet: A deep convolutional neural network approach for classification of rice varieties. *Expert Systems with Applications*, 235, Article 121214. <http://dx.doi.org/10.1016/j.eswa.2023.121214>.
- Erdogmus, P., & Kabakus, A. T. (2023). The promise of convolutional neural networks for the early diagnosis of the Alzheimer's disease. *Engineering Applications of Artificial Intelligence*, 123A, Article 106254. <http://dx.doi.org/10.1016/j.engappai.2023.106254>.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *29th IEEE conference on computer vision and pattern recognition* (pp. 770–778). <http://dx.doi.org/10.1109/CVPR.2016.90>.
- Iqbal, S., Khan, T. M., Naqvi, S. S., & Holmes, G. (2023). MLR-Net: A multi-layer residual convolutional neural network for leather defect segmentation. *Engineering Applications of Artificial Intelligence*, 126C, Article 107007. <http://dx.doi.org/10.1016/j.engappai.2023.107007>.
- Jardines, A., Eivazi, H., Zea, E., García-Heras, J., Simarro, J., Otero, E., et al. (2024). Thunderstorm prediction during pre-tactical air-traffic-flow management using convolutional neural networks. *Expert Systems with Applications*, 241, Article 122466. <http://dx.doi.org/10.1016/j.eswa.2023.122466>.
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521, 436–444. <http://dx.doi.org/10.1038/nature14539>.
- Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86, 2278–2324. <http://dx.doi.org/10.1109/5.726791>.
- Nguyen, M. T., Lin, W. W., & Huang, J. H. (2023). Heart sound classification using deep learning techniques based on Log-mel spectrogram. *Circuits, Systems, and Signal Processing*, 42, 344–360. <http://dx.doi.org/10.1007/s00034-022-02124-1>.
- Pachón-Suescún, C. G., Pinzón-Arenas, J. O., & Jiménez-Moreno, R. (2019). Detection of scratches on cars by means of CNN and R-CNN. *International Journal on Advanced Science Engineering and Information Technology*, 9, 745–752. <http://dx.doi.org/10.18517/ijaseit.9.3.6470>.
- Perry, J., & Fernandez, A. (2019). MinENet: A dilated CNN for semantic segmentation of eye features. In *17th IEEE/CVF international conference on computer vision workshop* (pp. 3671–3676). IEEE, <http://dx.doi.org/10.1109/ICCVW.2019.00453>.
- Qu, D., Huang, Z., Gao, Z., Zhao, Y., Zhao, X., & Song, G. (2018). An automatic system for smile recognition based on CNN and face detection. In *IEEE international conference on robotics and biomimetics* (pp. 243–247). IEEE, <http://dx.doi.org/10.1109/ROBIO.2018.8665310>.

- Rao, S. S., & Desai, S. R. (2022). Automatic dent detection in automobile using IR sensor. In V. Suma, X. Fernando, K. L. Du, & H. Wang (Eds.), *International conference on evolutionary computing and mobile sustainable networks* (pp. 501–511). Springer, [http://dx.doi.org/10.1007/978-981-16-9605-3\\_34](http://dx.doi.org/10.1007/978-981-16-9605-3_34).
- Shangzheng, L. (2019). A traffic sign image recognition and classification approach based on convolutional neural network. In *11th international conference on measuring technology and mechatronics automation* (pp. 408–411). <http://dx.doi.org/10.1109/ICMTMA.2019.00096>.
- Shanmugam, S., & Narayanan, R. S. (2024). An accurate estimation of hand gestures using optimal modified convolutional neural network. *Expert Systems with Applications*, 249, Article 123351. <http://dx.doi.org/10.1016/j.eswa.2024.123351>.
- Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In *36th international conference on machine learning* (pp. 6105–6114).
- Trethewey, M. (2000). Window and overlap processing effects on power estimates from spectra. *Mechanical Systems and Signal Processing*, 14, 267–278. <http://dx.doi.org/10.1006/mssp.1999.1274>.
- van Ruitenbeek, R., & Bhulai, S. (2022). Convolutional neural networks for vehicle damage detection. *Machine Learning with Applications*, 9, Article 100332. <http://dx.doi.org/10.1016/j.mlwa.2022.100332>.
- Zhou, Q., Chen, R., Huang, B., Liu, C., Yu, J., & Yu, X. (2019). An automatic surface defect inspection system for automobiles using machine vision methods. *Sensors*, 19(644), <http://dx.doi.org/10.3390/s19030644>.
- Zhu, Z., Li, D., Hu, Y., Li, J., Liu, D., & Li, J. (2021). Indoor scene segmentation algorithm based on full convolutional neural network. *Neural Computing and Applications*, 33, 8261–8273. <http://dx.doi.org/10.1007/s00521-020-04961-0>.