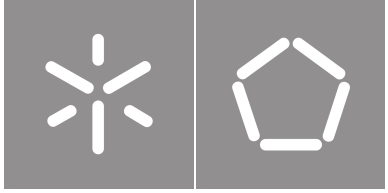


Universidade do Minho
Escola de Engenharia

André Rodrigues Soares

A deep-learning approach to detect scratches in vehicles

October, 2023



Universidade do Minho
Escola de Engenharia

André Rodrigues Soares

A deep-learning approach to detect scratches in vehicles

Integrated Master's in Informatics Engineering

Work developed under the supervision of:
João Miguel Lobo Fernandes
André Leite Ferreira

October, 2023

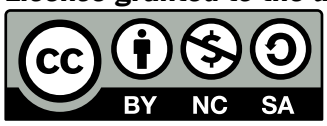
COPYRIGHT AND TERMS OF USE OF THIS WORK BY A THIRD PARTY

This is academic work that can be used by third parties as long as internationally accepted rules and good practices regarding copyright and related rights are respected.

Accordingly, this work may be used under the license provided below.

If the user needs permission to make use of the work under conditions not provided for in the indicated licensing, they should contact the author through the RepositoriUM of Universidade do Minho.

License granted to the users of this work



Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International

CC BY-NC-SA 4.0

https://creativecommons.org/licenses/by-nc-sa/4.0/deed.en

Acknowledgements

I would like to thank my mentors Professor Doctor João M. Fernandes and Professor Doctor André Ferreira for their availability, support and guidance during the development and writing of this thesis. I am grateful for the opportunity.

I would also like to thank my friends, in and out of university, for all the advice and friendship over these years and for their continued support.

And most importantly, I extend my deepest gratitude to my mother and father. Words cannot express my gratitude for all the sacrifice, love and support. Thank you from the bottom of my heart.

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the Universidade do Minho.

Braga, 20/12/2023

(Place) (Date)

Assinado por: **André Rodrigues Soares**
Num. de Identificação: 14081696
Data: 2023.12.20 12:14:19+00'00'

(André Rodrigues Soares)

"You cannot teach a man anything; you can only help him discover it in himself." (Galileo)

Resumo

Deteção de Eventos de Riscos em Veículos com Deep Learning

A deteção de danos na estrutura externa de veículos representa um desafio para os fornecedores de serviços de viaturas de aluguer, especialmente os serviços mais recentes de mobilidade em que a inspeção de danos não é realizada no final de cada aluguer. À medida que estes serviços se tornam mais populares e que mais pessoas deixam de ter necessidade de possuir um carro pessoal, espera-se que as empresas procurem formas de facilitar este tipo de inspeção de danos. Como tal, o objetivo principal desta dissertação é desenvolver uma solução que os fabricantes de automóveis poderiam potencialmente adotar. Esta solução envolve a criação de um algoritmo para detetar riscos em carros usando dados de áudio obtidos a partir de microfones.

Este estudo explora a utilização de Aprendizagem Profunda na análise de imagens, por exemplo espectrogramas, que são representativas de eventos de riscos. Para tal serão estudados e implementados métodos de transformação do sinal de áudio em imagens referentes a uma representação espectral do áudio e avaliar a capacidade de um algoritmo de Aprendizagem Profunda aprender a identificar este tipo de eventos. O algoritmo é treinado tendo em conta a enorme variedade de sons que podem ser captados pelo microfone, considerando os infinitos ambientes possíveis a que um carro pode estar sujeito, devido a uma condução diária.

Palavras-chave: Inteligência Artificial, Algoritmos, Aprendizagem Automática, Aprendizagem Profunda, Redes Neuronais

Abstract

A deep-learning approach to detect scratches in vehicles

The detection of damage to the external structure of vehicles poses a challenge for rental car service providers, especially in cases where damage inspection is not carried out at the end of each rental period. As these services become more popular as more people move away from the need to own a personal car, the expectation is that companies will be on the lookout for ways to facilitate this type of damage inspection. Therefore, the aim of this thesis is to create an urban mobility solution that car manufacturers could potentially adopt by developing an algorithm that detects scratches in vehicles, using audio data (obtained with microphones).

To achieve this goal, methods for transforming the audio signal into images representing the spectral characteristics of the audio are studied and implemented. Subsequently, the development and evaluation of a Deep Learning (DL) algorithm that can learn to identify such events is conducted. The algorithm undergoes training, using a vast array of sounds that a microphone can detect, within the countless everyday driving environments to which cars can be subjected.

Keywords: Artificial Intelligence, Machine Learning, Deep Learning, Algorithms, Damage Detection, Neural Networks

Contents

List of Figures xii

List of Tables xiv

Acronyms xv

1 Introduction 1

1.1 Context . . . 1
1.2 Objectives . . . 2
1.3 Key Contributions . . . 2
1.4 Document Outline . . . 3

2 State of the Art 4

2.1 Deep Learning . . . 4
2.1.1 Convolutional Neural Networks . . . 7
2.1.2 Recurrent Neural Networks . . . 10
2.1.3 Auto Encoders . . . 11
2.1.4 Attention . . . 12
2.2 Audio Signal Processing . . . 14
2.2.1 Time Domain Analysis . . . 14
2.2.2 Frequency Domain Analysis . . . 15
2.2.3 Time-Frequency Domain Analysis . . . 16
2.3 Related Work . . . 23
2.4 Metrics . . . 25

3 Development 28

3.1 Transfer Learning and Fine Tuning . . . 29
3.2 Data collection for scratch detection . . . 29
3.3 Labeling . . . 35
3.4 Dataset specifications and Time-Frequency Representations . . . 36
3.4.1 Dataset 1 specifications . . . 38

3.4.2 Dataset 2 specifications . . . 39
3.4.3 Dataset 3 specifications . . . 40
3.5 Metrics . . . 41
3.5.1 Establishing a threshold . . . 42
3.5.2 Image oriented metric . . . 43
3.5.3 Scratch oriented metric . . . 43
3.6 Exploring spectrogram mixing . . . 44
3.7 Exploring training set augmentations . . . 45
3.8 Networks choice and parameters . . . 47

4 Results and Discussion **50**
4.1 Dataset 1 . . . 50
4.1.1 Time-Frequency Representations . . . 50
4.1.2 Fourier Transform . . . 51
4.1.3 Data Augmentation . . . 51
4.1.4 Discussion . . . 51
4.2 Dataset 2 . . . 52
4.2.1 Data Augmentation & Batch Size . . . 52
4.2.2 Base Model fully trainable . . . 53
4.2.3 Best model results . . . 53
4.2.4 Discussion . . . 54
4.3 Dataset 3 . . . 55
4.3.1 Discussion . . . 55

5 Conclusion and Future work **57**
5.1 Conclusion . . . 57
5.2 Future Work . . . 59

Bibliographic References **60**

List of Figures

1 Artificial Neural Network (ANN) with multiple hidden layers 5
2 Neural Network before and after dropout [10] 6
3 Traditional transformations of data augmentation [11] 7
4 (a) Convolution and (b) max pooling operations. 8
5 LeNet5 Architecture [15] 8
6 AlexNet architecture: ImageNet 2012 winning CNN model [17] 9
7 VGG-Net architecture [17] 9
8 The inception module (adapted from [20]). 10
9 Recurrent Neural Networks with loop (Left) and its unfolded representation (Right) [21] . 11
10 Example of simple Autoencoder architecture 12
11 Self-Attention [22] - Color red represents the current word being fixated, blue represents memories. Shading indicates the degree of memory activation 13
12 The transformer - model architecture [31] 14
13 Example of Waveform 15
14 Example of Spectrum - frequency on the horizontal axis and magnitude in the vertical axis 16
15 Example of Spectrogram with linear frequency scale [42] 17
16 Example of Mel Spectrogram [43] 19
17 Wavelet transform over signal, from [44] 19
18 Example of Scalogram [45] 20
19 Constant-Q Transform spectrogram Example [49] 21
20 Cochleagram Example 21
21 Chromagram Example [54] 22
22 Mel Frequency Cepstral Coefficients (MFCC)s - outcome of spectrogram multiplication with the Discrete Cosine Transform (DCT) 23
23 Segment-based evaluation: Comparison of binary indicators between the reference and output annotation, adapted from [50] 27
24 Event-based evaluation with a collar of 100 ms, adapted from [50] 27
25 A general illustration of transfer learning/knowledge. The pre-trained Neural Network (NN) on the left is utilised as a base (without the classifier) for the NN on the right. 30

26 Example of label (in green) in internal BOSCH software 36
27 Window size and overlap length of the window function for STFT (adapted from [65]) . . 37
28 Different types of visual representations considered for the same scratch data 38
29 Example of scratches being pictured in log-Mel spectrograms 39
30 Example of different types of backgrounds events being pictured in log-Mel spectrograms 40
31 Example of more complex scratches being pictured in log-Mel spectrograms 41
32 Confusion Matrix for Binary Classification 42
33 Accumulative percentage of scratches by label (only for images with label higher than 0). 43
34 Scratch oriented classification 44
35 Example of log spectrogram, log-mel spectrogram and gammatoneagram mixing. 45
36 Example of log-mel spectrogram, chromagram, and gammatoneagram mixing. 46
37 Multiple type of augmentations used in this work 47
38 The architecture of the VGG16 model. 48

39 Prediction vs. Ground Truth Plot - Best Dataset 2 Model 54
40 Confusion matrix for (a) image-oriented and (b) scratch-oriented metrics for the best dataset 2 model. 54
41 Prediction vs. Ground Truth Plot - Best Dataset 3 Model 56
42 Confusion matrix for (a) image-oriented and (b) scratch-oriented metrics for the best dataset 3 model. 56

List of Tables

1 Damage events 31
2 Non-damage events. 33
3 Scratch events 34
4 Car models used in the data collection 34
5 Stationary and Moving data collection 34
6 Scratch and non-scratch events distribution in data collection 35
7 Difference between soft and hard labels for different spectrograms 36
8 Layout of all datasets considered 38
9 Distribution of images in train/test/validation sets featured in dataset 1 39
10 Distribution of images in train/test/validation sets featured in dataset 2 40
11 Distribution of images in train/test/validation sets featured in dataset 3 41

12 Results for different types of spectrograms 51
13 Results for different types of mel-scaled spectrograms 51
14 Results for different types of mel-scaled spectrograms 51
15 Results for different augmentations and batch sizes on dataset 2 53
16 Results for different augmentations and batch sizes with no pre-trained weights 53
17 Results for different augmentations and batch sizes on dataset 3 55

Acronyms

AE Autoencoder 11, 12, 23
AI Artificial Intelligence 1
ANN Artificial Neural Network xii, 4, 5, 10, 47
AUC Area Under the ROC Curve 26

CNN Convolutional Neural Network 3, 6, 7, 8, 9, 10, 11, 13, 15, 16, 24, 25, 29, 47, 48, 57, 59
CQT Constant-Q Transform 20, 22
CRNN Convolutional Recurrent Neural Network 11, 24, 25
CVAE Convolutional Variational Autoencoder 24
CWT Continuous Wavelet Transform 18, 19

DCASE Detection and Classification of Acoustic Scenes and Events 24, 25
DCT Discrete Cosine Transform xii, 22, 23
DFT Discrete Fourier Transform 15
DL Deep Learning ix, 4, 5, 6, 29, 57, 58
DNN Deep Neural Network 5, 6, 28

FN False Negative 25, 26, 41
FNN Feed Forward Neural Network 25
FP False Positive 25, 26, 41, 52
FPR False Positive Rate 26
FT Fourier Transform 15, 16, 22

GRU Gated Recurrent Unit 11, 24

LSTM Long Short Term Memory 11, 12, 24

MFCC Mel Frequency Cepstral Coefficients xii, 22, 23, 24, 25

ML Machine Learning 5, 28, 35

NN Neural Network xii, 3, 4, 5, 7, 8, 11, 23, 30, 49

R-CRNN Region-based Convolutional Recurrent Neural Network 24
RNN Recurrent Neural Network 5, 6, 10, 11, 13
ROC Receiver Operating Characteristic 26

STFT Short Time Fourier Transform 16, 18, 20, 21, 22, 37
SVM Support Vector Machine 25

TN True Negative 25, 41
TP True Positive 25, 26, 41
TPR True Positive Rate 26

WT Wavelet Transform 18

Chapter 1

Introduction

1.1 Context

Artificial Intelligence (AI) undeniably has, and will increasingly have, a significant impact on society across various domains. Deep Learning, in particular, has revolutionized 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 car rental services, which have gained increased attention as a viable alternative to vehicle ownership, particularly in large cities with well-developed public transportation infrastructure. Choosing to rent a car for occasional trips eliminates concerns related to maintenance costs, taxes, and insurance, providing a fast and accessible way of getting around a city. This makes it an attractive option for a large number of people, that put aside the otherwise usual demmand of car ownership.

From a business standpoint, it is evident that customers do not always take the best care of the rental vehicle while they are in full possession of the car, therefore, a comprehensive inspection for damages at the end of each rental period becomes necessary, particularly considering that no company driver is involved. 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 environments of the vehicle, equipping it with sensory capabilities to detect impacts that may lead to damages.

This dissertation is the result of a collaboration between BOSCH CAR MULTIMEDIA S.A., located in Braga, Portugal, and the University of Minho as an internship. The company is currently working on a product for a car group, intending to deploy it in a global fleet of vehicles as part of a mobility solution for city centers. This solution is designed as a car-sharing service available to the general public on demand, aiming to replace the existing car rental business with a more user-friendly, efficient system utilizing both

current and future technologies. The project aims to solve the damage inspection problem by leveraging deep learning techniques to create a system capable of identifying scratches within a vehicle equipped with sensory capabilities, by gathering audio data with microphones. Such a system could have a number of applications in the automotive industry. It promises to significantly reduce inspection costs, mitigate human errors in damage assessments, offer valuable insights about the car's condition to the driver, and provide real-time updates to car rental companies regarding the status of their rented fleet of vehicles.

1.2 Objectives

The aim of this dissertation is **to develop an algorithm to detect scratches in cars, using audio data (obtained with microphones).**

This is achieved through the transformation of audio data into a visual format, coupled with the deployment of advanced Deep Learning techniques for classification and detection.

In pursuit of the main objective, this work is guided by several key aims:

- O1: To develop a method for scratch detection in vehicles through the transformation of audio signals using neural networks. This step aims to check the feasibility of individually detecting scratches by feeding images to the networks under ideal, non-realistic conditions without the interference of background sounds.

- O2: To develop a method for scratch detection in vehicles that can handle more realistic scenarios by accounting for the polyphonic acoustic environment to which a vehicle is exposed.

- O3: To obtain a reliable and valuable algorithm that fulfills the proposed requirements by achieving a Matthews Correlation Coefficient (MCC) above 0.95, making it feasible for implementation in a real system used by consumers.

Throughout the development process, a constant analysis of the various results is essential. 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.

1.3 Key Contributions

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 Convolutional Neural Networks (CNNs).
- Use of the VGG16 CNN model for conducting all the development effort.
- The final version of the algorithm achieved MCC scores of over 0.90 for the best models, which are considered very good and close to be suited for industrial application.

1.4 Document Outline

This document is organized into the following chapters:

- **Introduction** : Presents the context, motivation, key contributions and objectives defined in the pipeline.
- **State of the Art**: Related search and relevant concepts and a statistic analysis made on the topics research.
  - Artificial Neural Networks - brief introduction to the basics of NN
  - Deep Learning - Study and research of techniques and detailed explanation on deep learning.
  - Audio Signal Processing - Study and research of techniques to visually represent digital audio signals
  - Related Work - Relevant studies and concepts related to deep learning techniques applied to audio detection with special focus on the use of visual representations.
- **Development**: Outline of the technical procedures involved in executing the proposed solution. Discussion of tools, methods, and strategies used in detail. Provides the reader with a comprehensive description of the dataset used, offering insights into the essential characteristics of the project's development.
- **Results and Discussion**: Outcomes of implementing the studied methods are thoroughly analyzed and discussed. It provides valuable insights into the different modules of the project, discussing it's achievements and thoughts process.
- **Conclusion and Future work**: Holistic overview of the entire project, summarizing the critical steps taken and the conclusions drawn from the results presented in Chapter 4. It includes a critical assessment of the project, highlighting its strengths and weaknesses and areas for improvement, possible alternative paths and suggestions for future developments.

Chapter 2

State of the Art

2.1 Deep Learning

To comprehend the state-of-the-art research in DL, it is essential to dive into the concept of ANNs. As the name implies, NNs are inspired by a simplified model of the human brain. ANNs consist of layers of neurons, which serve as the fundamental processing units of the network and possess the ability to learn. Each neuron computes a non-linear function of its inputs, assigning different weights to each input and transmitting the result to the subsequent layer. Similar to the development of a baby's brain as they interact with the environment, the connections between neurons in a neural network are strengthened or weakened as it processes input data. This iterative process, known as an epoch, involves two main phases, which are cycled through until a stopping criterion is met:

- **Forward phase:** Neurons are sequentially activated from the input layer to the output layer. Each neuron applies its weights and bias to the inputs and passes the result through an activation function, which generates the output. Activation functions are non-linear transformations that operate on a single number. Commonly used activation functions include Sigmoid, Tanh, and ReLU. [1].

- **Backward phase:** The network's output signal, obtained during the forward phase, is compared to the true target value in the training data. The discrepancy between the network's output signal and the true value produces an error that propagates backward in the network. This error is then used to modify the connection weights between neurons, aiming to reduce future errors.

This process of minimizing the loss or cost function by going back in the network and adjusting the weights and biases of the nodes is known as the back-propagation algorithm. This algorithm serves as the optimizer, determining how the weights are updated during training in each iteration. It iteratively minimizes

the loss, allowing the network to capture the desired distribution more effectively. Some commonly used optimizers include Stochastic Gradient Descent (SGD), Root Mean Square Propagation (RMSprop), and Adaptive Moment Estimation (Adam) [2].

ANNs date to the 1940s when McCulloch and Pitts proposed the first computational model of a neuron in the MP model [3]. Even though the MP neuron had no learning mechanism and limited capability it laid the foundation for DL and ANNs. Later in the 1957, Rosenblatt invented the perceptron algorithm, a binary classifier that had learning mechanisms [4]. The good performance of the perceptron inspired the research of shallow neural networks for many years, until 1969, Minsky and Papert pointed the limitations of the algorithm [5] and the trust in the field lied down, which triggered a NN winter season.

ANNs were revived in the 80s when Hopfield proposed the Hopfield Network which can be seen as a Recurrent Neural Network (RNN) and notably the Boltzmann machine proposed by Hinton in 1985 that showed the successful implementation of back-propagation in a NN [6]. After a slow start in the new century, ANNs received once again interest from the research community when Hinton put forward the concept of DL in 2006 [7]. The advancements of Hinton by developing Deep Belief Networks, which can be seen as a network with multiple hidden layers, were used by other researchers to train multiple types of deep networks and led to the popularization of the term Deep Learning that has since become a major area of research.

Although conventional machine-learning techniques were sucessfull in empowering many aspects of our lives they were limited in their ability to process natural data in their raw form which triggered an increased demand of a higher level of data processing [8]. In DL, the algorithm learns the true representation and feature of data by itself, there's no need of manual feature extraction. DL-based systems are developed from ANN, more particularly allied to the concept of Deep Neural Network (DNN). DNNs represent a feed-forward ANN with more than one hidden layer between the input and output layer, figure 1 pictures an example of an ANN with multiple hidden layers. In this sense DL as a subfield of Machine Learning (ML), enjoys of a stronger learning ability and better use of datasets for features extraction emerging as a champion in many fields, more particularly, in computer vision and natural language processing [9].

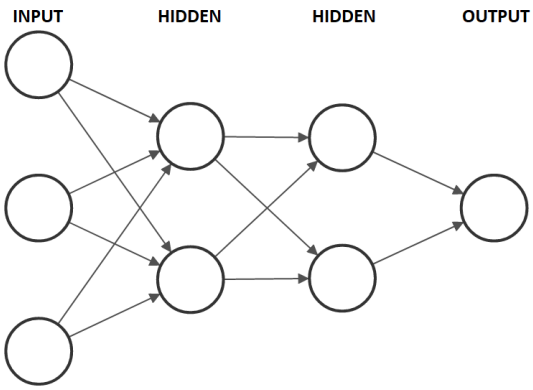


Figure 1: ANN with multiple hidden layers

A very important potential issue in Machine and Deep Learning algorithms is the possibility of overfitting where the model created performs well on training data but generalizes poorly on new data, that means that the algorithm learned the specific details of the training data rather than the overall patterns that are common to the whole dataset. If the patterns learned from the data don't apply to new examples it can not be said that the model actually learned anything particularly useful. Usually feeding more training data to the model is the most straightforward way to fix this but unfortunately this is not always viable. In order to overcome this problem in DL we use what's called regularization techniques. Regularization is a technique that make simple modifications to the learning algorithm so that the model generalizes better and have a better performance in unseen data. The most commonly types of regularization are L1 and L2 which add a new term known as regularization term to the general cost function.Another common solution is early stopping, where, to stop the algorithm from overfitting, the training process is stopped when the testing error on our test dataset starts increasing. Besides that it is worth mentioning dropout, a technique used to regulate the network by randomly skipping connections. Dropout gives big improvements on many benchmark tasks and set new records for speech and object recognition [10]. An example of a neural network skipping connections with the dropout tecnique is pictured in figure 2.

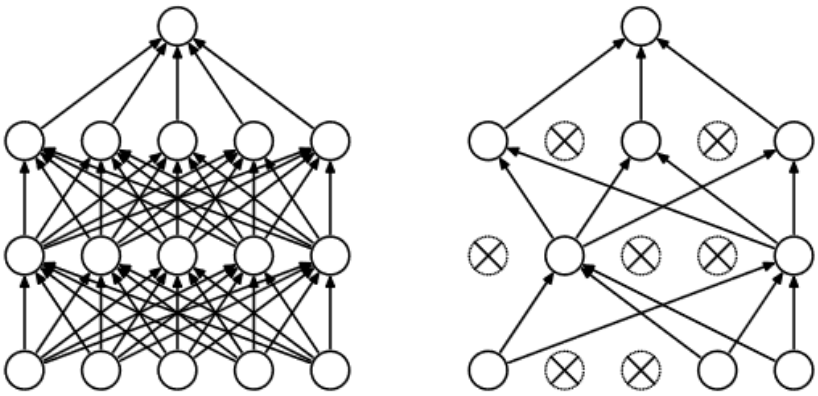


Figure 2: Neural Network before and after dropout [10]

Data augmentation is another widely used technique for regularization [11]. It addresses the common challenge of insufficient data, particularly in tasks related to image and video classification. Data augmentation provides a reliable method to expand the dataset size by applying transformations to the input data. In the context of image data augmentation, common practices include geometric and color transformations. These transformations involve operations such as image reflection, cropping, translation, and adjusting the color palette. Figure 3 shows traditional data augmentation transformations involving zooms, distortions and rotations.

DNNs structure have higher complexity therefore it takes more time to train them, although carrying a greater capability of learning. The most popular DNN approaches are CNNs, very useful in image analysis, being able to extract feature from the input images; and RNNs, networks with cycles within its connections

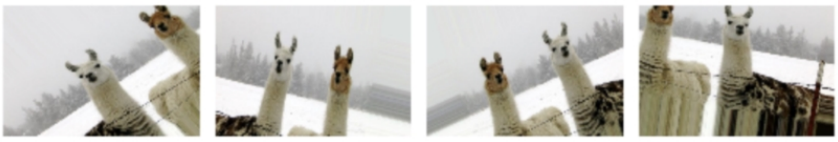


Figure 3: Traditional transformations of data augmentation [11]

that have an internal state (memory) and usually used to work with sequential or time series data as input data.

2.1.1 Convolutional Neural Networks

CNNs are loosely based on the manner in which all mammals perceive the world around them, utilizing a hierarchical series of feature recognition. 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 color image or an audio spectrogram, the computer perceives the image as a matrix of pixels with height, width, and depth, where each color channel (RGB) corresponds to a 2D array. Therefore, a 24x24x3 image matrix refers to a 24x24 image size with 3 color 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 convolutional networks 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 [12], segmentation of biological features [13], and face detection [14].

The architecture of a ConvNet consists of two fundamental types of layers: convolutional layers and pooling layers [8].

- **Convolutional layer:** The basic unit of a Convolutional Network where we 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 NN. The convolution operation involves applying a filter to the entire image, extracting relevant features. Figure 4(a) shows an example of an convolution operation: applying the kernel to the small green patch in the input matrix, the result obtained with $4 * 0 + 3 * 1 + 0 * 2 + 1 * 3$ 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 network to utilize fewer parameters. This makes the neural network 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. [1]. Figure 4(b) shows an example of the max pooling operation applied to a matrix where the maximum value, captured by a 2 by 2 filter with strides of 2 by 2, is passed to the output matrix.

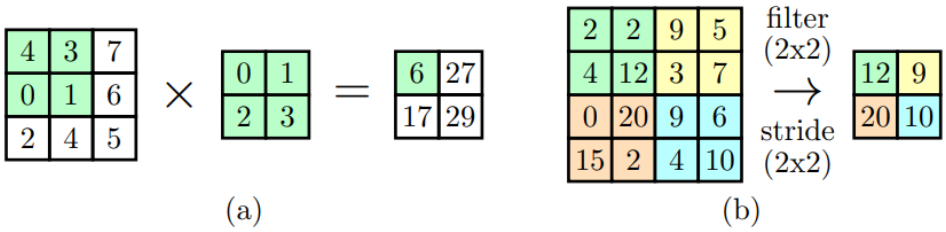


Figure 4: (a) Convolution and (b) max pooling operations.

After the feature extraction phase in the convolutional and pooling layers, the network utilizes fully-connected layers. These layers are simple feed-forward NN 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, regularization techniques such as dropout are often implemented at this stage.

2.1.1.1 CNN architectures

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 network 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).

One of the pioneering CNN architectures is LeNet5, introduced by LeCun in 1998 [15] for handwriting recognition. It comprised seven layers and was the first widely known deep convolutional neural network architecture that can be seen in figure 5.

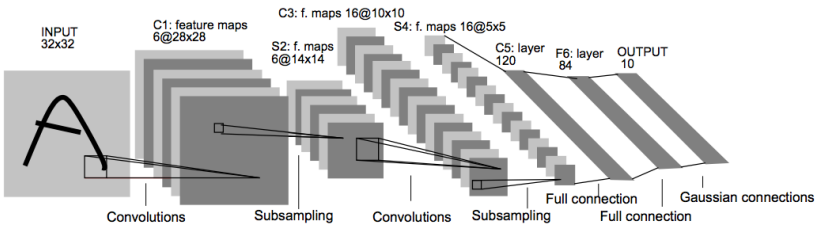


Figure 5: LeNet5 Architecture [15]

Another significant breakthrough occurred in 2012 [16] when AlexNet demonstrated the power of deep CNNs in significantly improving detection performance. AlexNet achieved a top-5 error rate of 15.3%, which was a remarkable 10 percentage points advantage over the second-place contestant that did not use CNNs. This CNN consisted of seven hidden layers, 60 million parameters, and introduced several techniques that have become common in other architectures, such as max pooling and GPU implementation for accelerated training. One notable difference from LeNet was the adoption of the Rectified Linear Unit (ReLU) function as the activation function for neurons, replacing hyperbolic or sigmoid functions. ReLU has become the most commonly used activation function in CNNs due to its faster convergence during training. Since then, CNNs have been extensively applied to various computer vision problems. Figure 6 shows the AlexNet architecture.

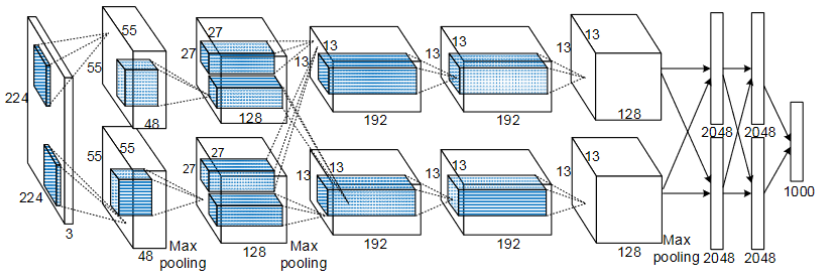


Figure 6: AlexNet architecture: ImageNet 2012 winning CNN model [17]

AlexNet, along with the VGG networks, which won the ILSVRC2014 classification and localization competition, are often used as pre-trained image classifiers. The VGG networks [18], shown in figure 7, popularized the use of smaller filters, such as 3x3, instead of larger ones (e.g., AlexNet used 11x11 filters). VGG networks have shown good overall performance in multiple contexts.

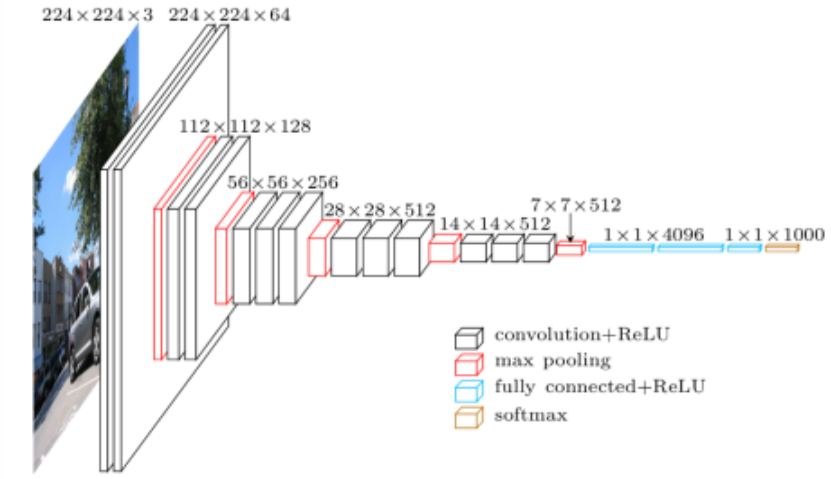


Figure 7: VGG-Net architecture [17]

There are other notable CNN architectures that have made significant contributions to the field of

computer vision, including ResNet [19] and GoogLeNet [20]. GoogLeNet gained fame after winning the 2014 ImageNet ILSVRC challenge. This architecture introduced the concept of an inception module, as shown in figure 8, which consists of parallel convolutional layers with different filter sizes. Unlike previous architectures where the filter sizes were manually determined by developers, in GoogLeNet, the algorithm decides the appropriate sizes. To address the computational resources required for the alternate paths, GoogLeNet utilizes 1x1 convolutional layers to reduce the number of features before passing through the convolutional layers with larger filters [20].

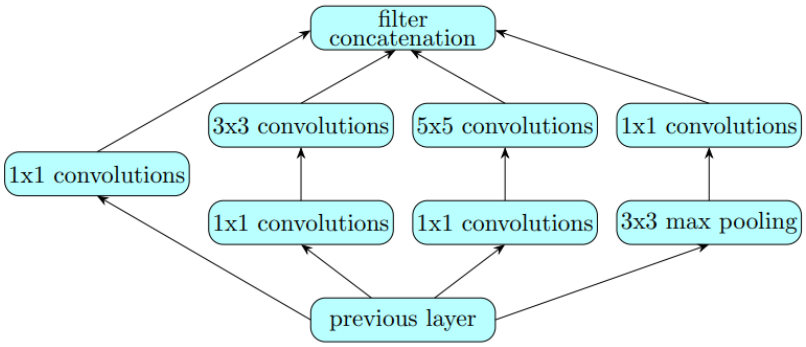


Figure 8: The inception module (adapted from [20]).

ResNet, developed by Kaiming et al. [19], introduced the concept of shortcut connections across layers. These non-weighted connections allow the network to learn the identity function of the input if the extra parameters are unnecessary for the specific problem. This innovation helps alleviate the problem of vanishing and exploding gradients. The vanishing gradient problem happens because as ANNs become deeper the gradients used to update the weights of the network during training keep getting smaller throughout the network slowing the training to a crawl. On the other hand, the exploding gradient problem arises when gradients become excessively large, potentially causing overflow issues. With shortcut layers, ResNet enables the construction of deeper networks that were previously challenging to train effectively.

2.1.2 Recurrent Neural Networks

Feed-forward Neural Networks like CNNs process frames independently, meaning that a prediction is based only on the input at the same moment without using any context from the previous time step. Recurrent neural networks is a type of ANN that uses sequential or time series data as input data and where we can find direct cycles in its connections of the network like it can be seen in figure 9. These cycles allow networks to have a characteristic that differentiates them from regular networks - they take information from prior inputs to influence the current input and output therefore acting like a internal state memory, very useful when analyzing data that has a correlation in time.

The initial version of RNNs, known as Vanilla RNNs, faced challenges with storing information from long sequences due to the vanishing or exploding gradient problem, which arises when gradients become

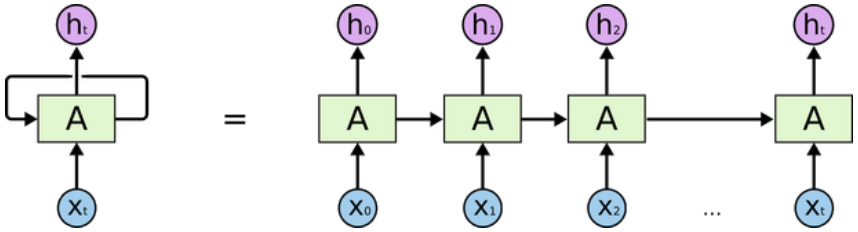


Figure 9: Recurrent Neural Networks with loop (Left) and its unfolded representation (Right) [21]

extremely small or large during training. This issue was addressed with the introduction of Long Short Term Memory (LSTM) networks [22]. LSTM is a variation of RNNs that introduced gating mechanisms and explicit memory cells. The gates control the flow of information, allowing LSTM networks to store and access relevant information from previous time steps until a deletion instruction is triggered [23]. Another alternative to LSTM networks is Gated Recurrent Units (GRUs) [24]. GRUs also utilize gating mechanisms but have a simpler architecture compared to LSTM networks. They perform better than LSTM networks in certain scenarios, particularly when working with small datasets [24], and have fewer parameters due to the absence of an output gate.

RNNs are often used in combination with CNNs to create hybrid models known as Convolutional Recurrent Neural Networks (CRNNs). These models are commonly applied in tasks such as sound event detection, where convolutional layers serve as feature extractors, capturing high-level features from time-frequency representations like spectrograms, while the recurrent layers provide temporal context learning for the extracted features. The combination of CNNs and RNNs in CRNNs leverages the strengths of both architectures to improve performance on sequential data analysis tasks.

2.1.3 Auto Encoders

Autoencoders (AEs) emerged as a way of reducing the dimensionality of data [25]. They are trained to learn efficient data codings in an unsupervised manner. In a autoencoder the number of neurons in the input and output layers are the same, therefore, by feeding an image to the network banks on the fact that the output has not only the same size but represents the same image, essentially the NN acts like a copying machine. The aim is to learn a representation of a given dataset by training the network to ignore useless features. An AE can be seen as two different NNs, the encoder and the decoder, as pictured in figure 10.

The encoder's role is to map the input data from a high-dimensional space to a lower-dimensional space. In an AE, there is a bottleneck layer in the middle, where the number of neurons is significantly reduced compared to the input and output layers. This forces the network to find a compact representation of the data using a smaller number of neurons, focusing on the essential features. The output of the encoder is not expected to be an exact copy of the input, but rather a close approximation that captures the important information. The decoder, on the other hand, aims to reconstruct the original input data using the

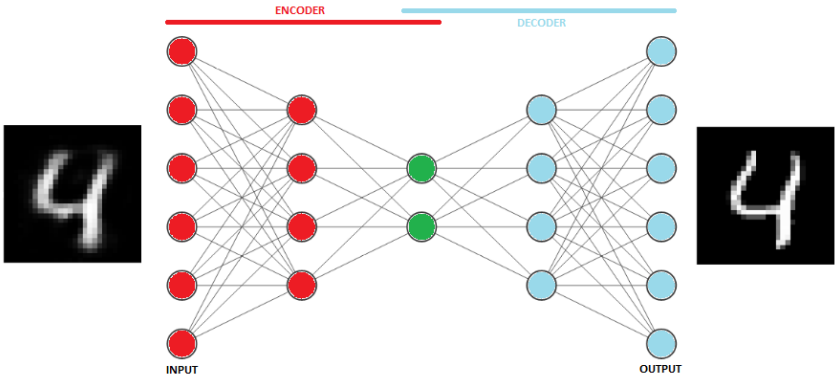


Figure 10: Example of simple Autoencoder architecture

encoded representation from the bottleneck layer. It attempts to reverse the encoding process and generate a higher-dimensional output based on a lower-dimensional input [25]. The quality of the reconstruction is evaluated using a reconstruction error, which measures the difference between the original input and the output of the decoder. By minimizing this reconstruction error during training, the network learns to effectively represent the data in a lower-dimensional space.

One advantage of AEs is their ability to create sparse representations of the input data. This means that only a subset of the neurons in the encoder layer is activated, leading to a more compact and meaningful representation. Autoencoders can be applied to various tasks such as image compression [26], image denoising [27], audio anomaly detection [28], and are commonly used as feature extractors [29] by utilizing only the encoder part of the network.

2.1.4 Attention

The Bahdanau Attention Mechanism [30] was introduced to help with neural machine translation tasks. The encoder-decoder model was encoding entire input sentences into fixed length context vectors that the decoder would use to generate a translation. The use of this fixed length context vectors turned out to be a bottleneck in performance especially when dealing with long sentences. Given that reading an entire paragraph before translating it's hard, the main idea behind attention is that the reading is made "as we go"and that for each word in the decoder we focus our attention on a few words at a time, this is done by weighting each of the output activations of the encoder. In the LSTM network paper [22] a self-attention mechanism is used to do machine reading. In figure 11 it can be seen how self-attention enables us to learn the correlation between the current words and the previous part of a sentence.

The self-attention technique highly improved the quality of machine translation systems by allowing them to focus on the relevant parts of the input sequence as needed. The concept lies in the functionality of looking to the context of other tokens in the same sentence therefore representing a relationship between words which translates in a better understanding of them. Bahdanau et al. noticed a big improvement with sentences longer than 25 words in this type of architecture compared to the basic encoder decoder

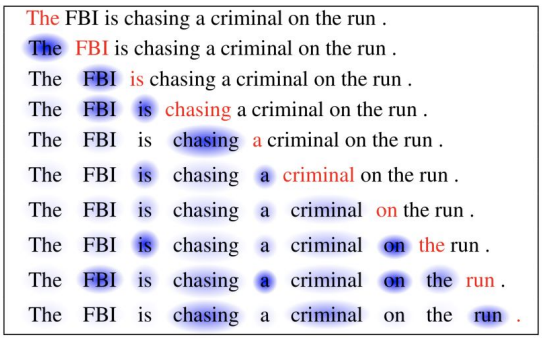


Figure 11: Self-Attention [22] - Color red represents the current word being fixated, blue represents memories. Shading indicates the degree of memory activation

RNN [30].

2.1.4.1 Transformers

The introduction of the transformer by Vaswani [31] ended up being the beginning of the end of RNNs as the state-of-art tool for machine translation. The sequential nature of RNNs means that the data is processed in order, sentences are processed word by word. This shortcoming is solved in transformers architectures by being able to run computational tasks in parallel, the model is entirely built on the self-attention mechanism and no recurrence is needed to encode sequential data. The high-level architecture of the model is shown in Figure 12.

The transformer adopts the encoder decoder architecture. The encoder is composed by a stack of identical layers, each one can be broken down in two sub-layers, the first a multi-head self-attention mechanism, and the second a simple feed forward network. The decoder is similar but employs a third sub-layer which performs multi-headed attention over the encoder's output and previous decoder attention layer. The removal of recurrent networks removes the model's ability to keep track of what the position of an input is in the sequence. The output from the decoder is fed to a linear layer with a softmax activation function producing a vector of probabilities. This vector represent what the next part of the sequence is most likely to be. By using positional encodings for both the input to the encoder and the decoder one can resolve the inability to keep track of positions.

Transformer-like architectures became more prominent in the past few years and are now used in computer vision since the introduction of the vision transformer (ViT) [32]. Vision transformers split images into patches, like the sequence of words when using transformers to text, and includes positional embedding to input to the encoder. In large-scale datasets they can already outperform state-of-the-art CNN based methods [33].

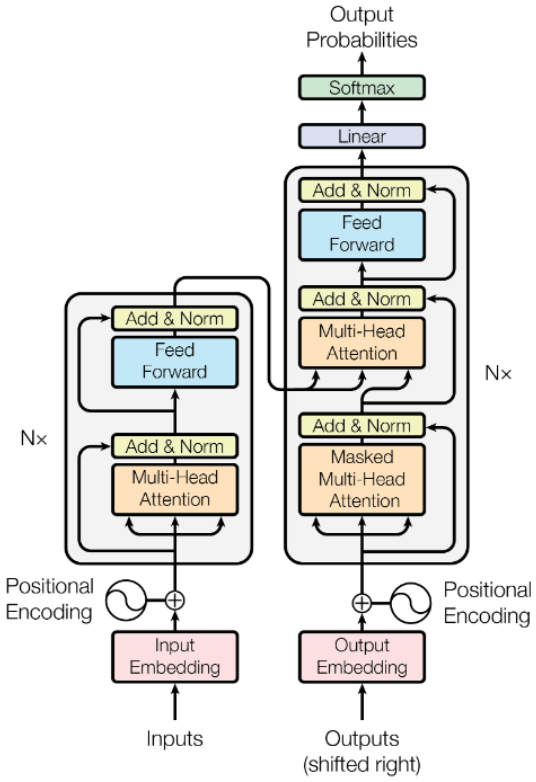


Figure 12: The transformer - model architecture [31]

2.2 Audio Signal Processing

Machine and Deep Learning have demonstrated excellent results in understanding the environment through vision. However, another crucial aspect of human perception is audio processing, which involves extracting meaningful information from the auditory environment. Machine and Deep Learning for audio tasks are used in speech and musical instrument recognition, event classification and detection, and even audio and music generation. Sound is a vibration that propagates as an acoustic wave, produced by the vibration of an object. These vibrations cause air molecules to oscillate and collide with each other, resulting in changes in air pressure in the surrounding region. This creates a wave that travels through space, typically through air, as the medium. To represent sound digitally, the analog signals are sampled at fixed intervals of time, converting the continuous signal into a discrete series of numbers that can be processed by our models. This process of sampling and quantifying the information is called Analog to Digital Conversion (ADC). The sample rate refers to the number of samples taken per second. For example, a 10-second music clip with a common sampling rate of 44.1 kHz would have 441,000 samples.

2.2.1 Time Domain Analysis

Time domain analysis focuses on studying physical signals that exhibit variations over time. A complex sound can be represented using a waveform, which shows how the signal's amplitude changes over time.

In the context of audio signals, the waveform represents the loudness of the sound as a function of time. Visualization of a waveform is shown in Fig. 13.

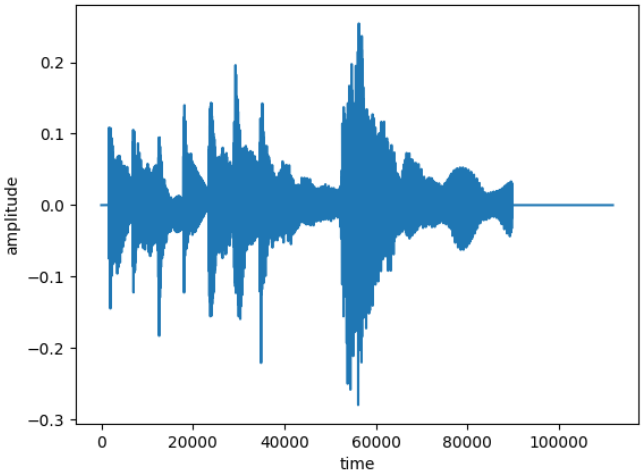


Figure 13: Example of Waveform

Some deep neural networks attempt to learn the underlying features of waveforms for classification tasks [34] [35]. However, time-frequency spectral features are more commonly used and will be discussed in more detail in further subsections.

2.2.2 Frequency Domain Analysis

Waveforms alone, as seen in 2.2.1, do not capture one of the most important characteristics of sound: frequency. Therefore, waveforms are less discriminative as representations. Alternatively, sound can be represented using a spectrum, pictured in figure 14. Spectral data is displayed in the frequency domain and is obtained by applying a Fourier Transform (FT) [36] to the time domain waveform. In the case of digitized signals, the discrete version of the Fourier Transform, known as the Discrete Fourier Transform (DFT), is used. By transforming the signal from the time domain to the frequency domain using the Fourier Transform, we obtain a spectrum that represents the frequency distribution of the signal. Essentially, we capture a picture of the sound and analyze the frequency components present in it, this is all frequencies that make up a sound.

Frequency-domain analysis has been shown to be more accurate than raw time-domain signals for CNNs-based classification tasks [37]. Despite their separate representations, these two categories of audio features complement each other. Our focus will be on time-frequency representations, such as spectrograms, that combine both time and frequency information.

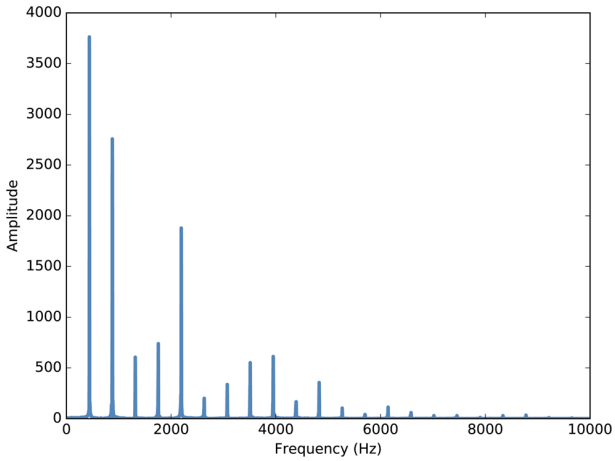


Figure 14: Example of Spectrum - frequency on the horizontal axis and magnitude in the vertical axis

2.2.3 Time-Frequency Domain Analysis

Visual displays of audio signals using various time-frequency representations, such as spectrograms, provide a rich representation of the temporal and spectral structure of the original signal [38]. This is particularly useful considering the popularity of CNNs for computer vision tasks that receive images as input. There are multiple ways to represent an audio signal in the time-frequency domain, including spectrograms with linear or logarithmic scales, mel-spectrograms, constant q-transforms, scalograms, and cochleagrams. These representations are commonly used in building Deep Learning algorithms for audio processing tasks.

2.2.3.1 Spectrogram

The spectrogram is a representation of a signal that displays signal strength over time at various frequencies. Given an audio signal, its set of sampled points denoted as $x(n)$, in order to generate a spectrogram we need to split the original signal into short-time frames with equal length in order to apply the Short Time Fourier Transform (STFT). This basically consists in sliding a window through the signal applying the FT to each frame. We can think of a spectrogram as bunch of FTs stacked on top of each other. A frame contains N samples which determines the so called frame length and they usually overlap each other over 50% of the previous/next frame. To each frame is applied a window function $w(n)$, usually the widely known Hann window, in order to overcome spectral leaks through sidelobes. Then, the Fourier Transform is applied to each short-time frame x_α:

$$X_\alpha(k) = \sum_{n=0}^{N-1} x_\alpha(n) w(n) e^{-i2\pi n \frac{k}{N}} \tag{2.1}$$

In order to visualize the spectrogram we must calculate the power spectrum density by taking the squared magnitude of the frequency spectrum. In each short-time frame, its power spectrum density is denoted as $Pa(k)$, which is given as follows:

$$P_\alpha(k) = |X_\alpha(k)|^2 \tag{2.2}$$

Finally, to get closer of the way humans perceive sound we need to move the amplitude from a linear representation to a logarithmic representation. This is obtained by transferring the power into the decibel scale, which is a unit measuring the intensity of a sound. Decibel scale is inspired by human's perception of loudness. The human ear has a large dynamic range in sound reception, the ratio between the sound intensity that leads to permanent damage and the limit of smallest audible sound is close to 1 trillion, which is an enormous measurement range. To easily express this range, and conveniently measure an exponentially changing sound loudness over time, the base 10 logarithmic scale is motivated.

$$P_\alpha^{dB}(k) = 10log_{10}(P_\alpha(k)) \tag{2.3}$$

Spectrograms are two-dimensional graphs that visualizes sound's frequency spectrum as a function of time, with a third variable represented by color. The brighter the color the higher the amplitude or the power of a certain frequency at a particular time. In a spectrogram, not only can we identify where the energy is distributed over frequency, but also see how energy levels vary across time. The conventional spectrogram representation is still used in many areas such as speech emotion recognition [39], classification of instrumental audio [40] and sound event detection [41]. One example of a spectrogram is shown in figure 15.

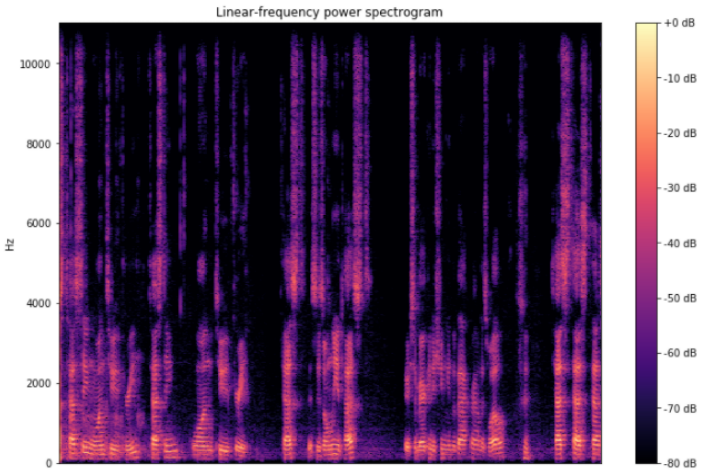


Figure 15: Example of Spectrogram with linear frequency scale [42]

2.2.3.2 Mel Spectrogram

Mel spectrograms are often the feature of choice to train Deep Learning Audio algorithms, consistently performing well across different type of situations when compared to other representations [38]. This type of spectrogram is obtained by applying the Mel filter banks to a spectrogram, converting the frequency

(Hz) scale into the Mel scale. This non-linear transformation is motivated by human's auditory system. A human will notice the difference between 500 and 1000 Hz, but will have a hard time telling the difference between 5,000 and 5,500 Hz even though the distance between the two pairs are the same. This makes us better at detecting differences in lower frequencies than higher frequencies. The Mel (m) scale is transformed from the frequency (f) scale by:

$$m(f) = 2595log_{10}(1 + \frac{f}{700}) \tag{2.4}$$

And can be transformed back by:

$$f(m) = 700(10^{m/2595} - 1) \tag{2.5}$$

Generating and calculating the Mel filter bank is a multi-step process:

1. Given the maximum and minimum frequency of the sampled signal we calculate the corresponding maximum and minimum Mel value
2. Specify the number of Mel filters N and create N + 2 linearly spaced points between the maximum and minimum Mel value
3. Convert the points back to the frequency scale based using equation 2.5
4. Round points to the nearest frequency bin
5. Finally create the triangular Mel filters

Calculating the Mel filter bank results in a matrix that can be multiplied with the power spectrum obtained in 2.2 originating the Mel spectrogram. Transforming the power into decibel scale like in 2.3 results in the log-Mel spectrogram. Figure 16 shows an example of a mel scaled spectrogram.

2.2.3.3 Scalogram

Scalograms are another spectro-temporal way to represent a signal, they are similar to spectrograms but while the latter decomposes the signal using sinusoidal functions, scalograms are obtained with the Continuous Wavelet Transform (CWT) which uses sets of wavelets represented by the Wavelet Transform (WT) [44]. The STFT fails to realize that different frequencies become apparent at different scales and the CWT takes advantage of this fact.

The CWT works just like the STFT where a window function is used to extract signal segments. In this type of transform the window is called a wavelet. Wavelets are a oscillation that is localized in time, unlike the window in STFT, the wavelet can not only be translated, but also dilated or contracted depending on the wavelet scale, figure 17 pictures a wavelet with different scales. Dilating a Wavelet increases the CWT's sensitivity to long time-scale events, and contracting it increases its sensitivity to short time-scale

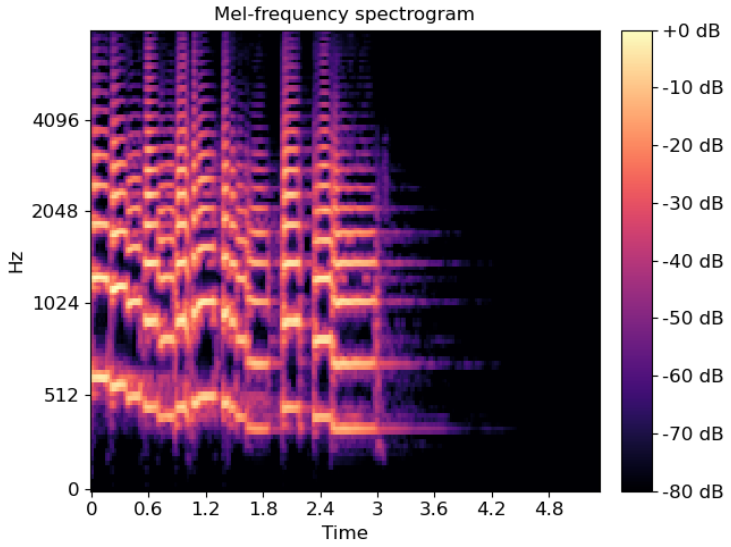


Figure 16: Example of Mel Spectrogram [43]

events, this makes the CWT ideal for analyzing non-stationary and transient signals at multiple scales [44]. Considering C as the transformation, a the scale factor, b the translation time, ψ the mother wavelet function and $x(t)$ the input signal in its time domain representation, the CWT is represented by [44]:

$$C(a,b) = \frac{1}{\sqrt{a}} \int \psi_{a,b}\left(\frac{t-b}{a}\right) x(t) dt \tag{2.6}$$

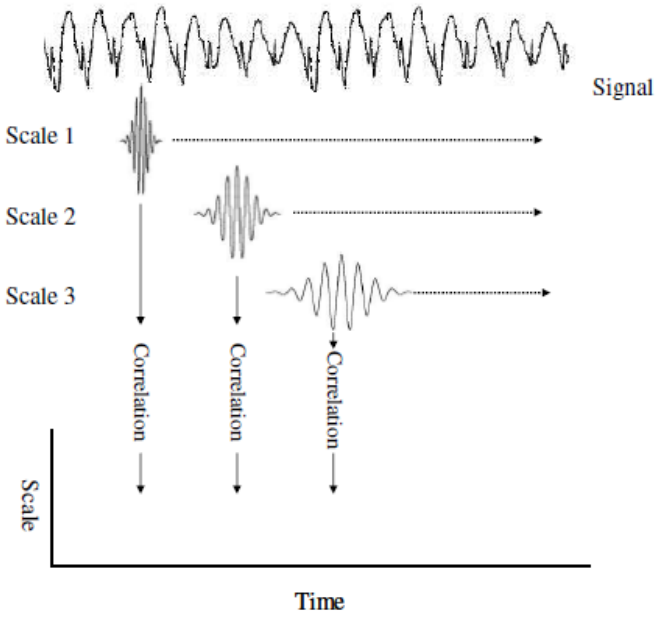


Figure 17: Wavelet transform over signal, from [44]

The CWT maps the signal into a function $C(a,b)$ that can be used to determine the similarity between $x(t)$ and a wavelet scaled by a at given time b, basically computing how much of a wavelet is in a signal.

There are multiple types of wavelets, the bump and morse wavelet are some of the most used. Scalograms are a time–frequency plot of the correlation between the signal and the scaled wavelets. Figure 18 shows an example of a scalogram.

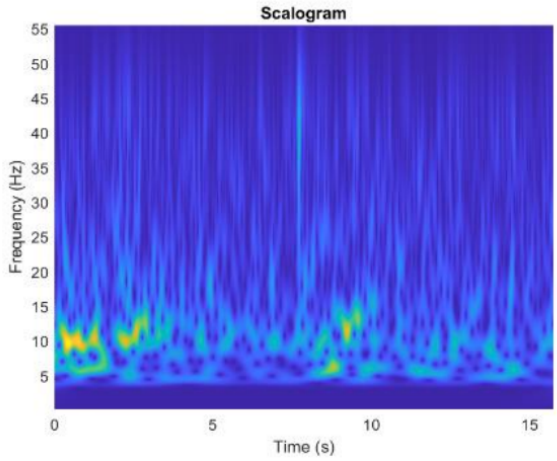


Figure 18: Example of Scalogram [45]

2.2.3.4 Constant-Q Transform

The Constant-Q Transform (CQT) is a variation of the STFT, in other words, a type of wavelet transform, originally introduced to map the western musical scale [46]. It's similar to the STFT but with geometrically spaced frequency bins. In the STFT we are faced with constant frequency resolution consequential from the constant window size. The CQT offers a better spectral resolution at lower frequencies and better temporal resolution at higher frequencies which makes it more suitable for music and speech as a time-frequency representation [47]. Figure 19 depicts a constant-q transform spectrogram.

The CQT of a time-domain signal $x[n]$ is defined as [48]:

$$X[k] = \frac{1}{N[k]} \sum_{n=0}^{N[k]-1} W[k,n]x[n]e^{-jw_k n}, \tag{2.7}$$

where $X[k]$ is the CQT coefficient for k-th frequency bin, $W[k,n]$ is the time-domain window for k-th bin with duration $N[k]$ and $w_k = \frac{2\pi Q}{N[k]}$, where Q is the (constant) Q factor of the filter banks.

The CQT spectrogram and scalogram are representations capable of having different time or frequency resolutions. This can be an advantage since different types of sounds may benefit from different resolutions. A short sound such as a gun shot benefits from a high time resolution but a bird singing may benefit from a higher frequency resolution [50].

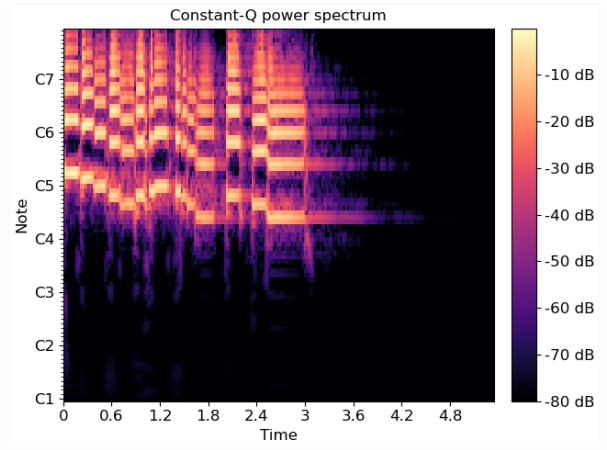


Figure 19: Constant-Q Transform spectrogram Example [49]

2.2.3.5 Gammatone-like spectrogram

The gammatone filters proposed by Roy Patterson et al. [51] are a signal-processing approximation inspired by the cochlea. In the ear the frequency subbands get wider for higher frequencies and spectrograms have constant bandwidth across all frequency channels which means that can be a unreliable representation. Alternatively to the conventional spectrogram, a time-frequency representation based on gammatone filterbanks can be created. The so called Gammatonegram (or Cochleagram) can be described as a visual form of representing the energy of a STFT based signal with these filterbanks [52] and example can be seen in figure 20.

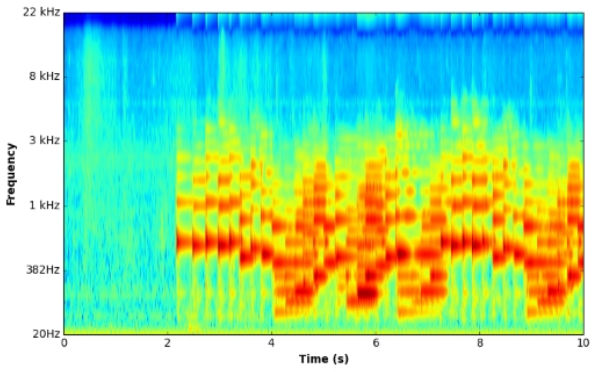


Figure 20: Cochleagram Example

2.2.3.6 Chromagram

The Chromagram is a time-frequency or time-chroma distribution which represents the spectral energy at each of the 12 chroma or pitch classes [53]. Two notes with frequencies in a ratio equal to any power of two (e.g., half, twice, or four times) are perceived very similar by humans. The sinusoids 110, 220 and 880 Hz all can be grouped under the chroma/pitch class A even though they differ in their absolute

frequencies. An octave is the interval between one note and another with half or double its frequency and the pitch class is a group of all the pitches of the same type regardless of the octave they are in [53].

The chroma spectrum S(c) maps the strength of a signal with respect to a chroma value. Similarly to the standard spectrum obtained by the FT that can be extended to a time-frequency distribution s(t,f) a time-chroma distribution s(t,c) can be defined through some function G:

$$s(t, c) = G(s(t, f)) \tag{2.8}$$

where $f = 2_{c+h}$ with respect to the value of chroma c and height tone h that indicates the octave the pitch is in. The chromagram can be computed from spectrograms, comparison of a long-window STFT chromagram with CQT chromagram is shown in figure 21.

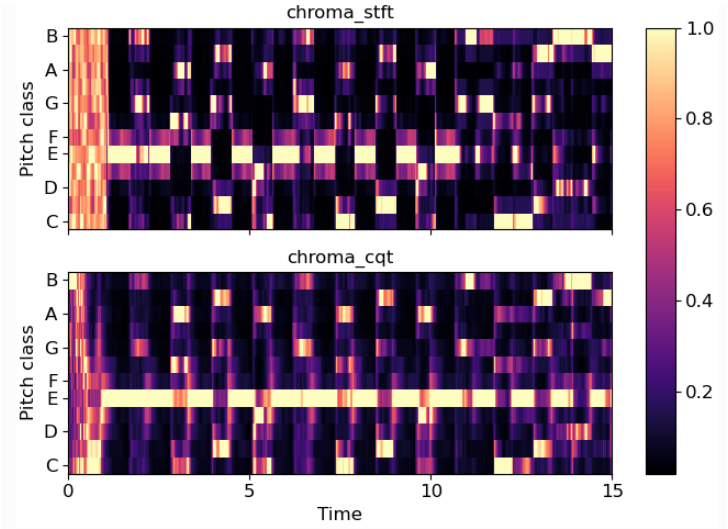


Figure 21: Chromagram Example [54]

2.2.3.7 MFCC

MFCC is a feature extraction technique that's also based on human hearing perceptions, commonly used in audio classification particularly in automatic speech and speaker recognition [55]. The MFCC calculates the logarithm energy of the mel filter bank using the DCT and can be seen in figure 22. The MFCC extraction process is as follows [55]:

- Pre-Emphasis: Passes the signal through a filter emphasizing higher frequencies
- Framing: Segmentation of the block in smaller frames, the signal is divided into frames of multiple samples.
- Hamming Hamming Windowing to keep continuity of the signal.
- Fast Fourier Transform (FFT): Convert each frame from the time domain into the frequency domain.

- Triangular band pass filters: Multiply the magnitude frequency response with a set number of triangular band passes to achieve a smooth magnitude spectrum, reducing the size of features in the process.
- Discrete Cosine Transform (DCT): Apply DCT to convert the log mel spectrum into the time domain. Result of conversion is the MFCC.

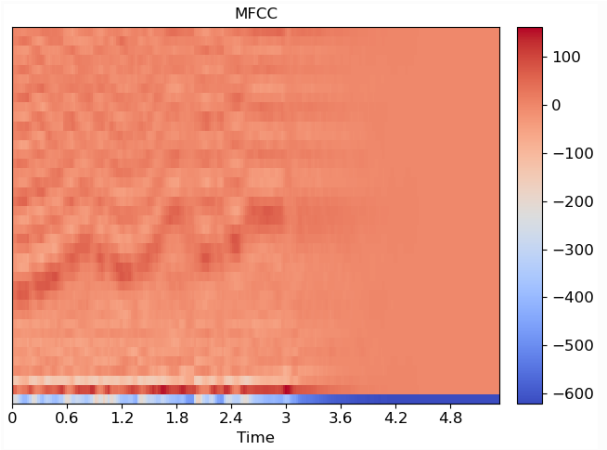


Figure 22: MFCCs - outcome of spectrogram multiplication with the DCT

2.3 Related Work

The first stage of the development of a scratch event detection algorithm consisted of a in-depth research on the field of NNs and digital audio spectral representations. Then, a study on existing approaches was carried out following a guideline that tried to connect NNs, spectrograms or similar representations and scratch detection in cars. Unfortunately, no articles were found with the three subjects in hand, and a broader perspective had to be followed with special focus on Audio Event Detection using NNs.

Robert Muller et al [56] proposed a system that intends to detect fraudulent behavior in factory machinery by focusing on detecting anomalous sounds. The authors challenge usual approaches to unsupervised anomaly detection based on deep autoencoders. The study focus on extracting features using neural networks that were pretrained for image classification tasks by transforming the raw audio in mel-spectrograms and combine those feature with traditional anomaly detection models. The pretrained feature extractors are Alexnetv3, RestNet18, ResNet34 and SqueezeNet and two AEs trained from scratch are also used for comparison purposes since it's the predominant approach to similar problems. The results indicate that image transfer learning is more effective for anomalous sound detection than autoencoders trained from scratch and that the ResNet architectures are the superior feature extractors. Likewise, the Gaussian Mixture Model and One-Class Support Vector Machine yielded the best performance as anomaly detection models.

Sergey A. Romanov et al. [57] came up with a acoustic event detection system capable or recognizing more than 10 types of sound events (e.g. scream, siren, sneeze, etc). Due to lack of a large amount of data the authors settle for a VGGish network pretrained on AudioSet by Google and the feature extraction is made using log-mel spectrograms. This model converged much faster than initial model attempts based on Mobilenet and Resnet. Even though the problem is related to audio detection, the followed approach is by training binary classifiers for each class of sound and then slide through a sample classifying small patches. An input sample is divided in frames and each frame is classified, compared to a threshold and if it's found a match, then it's decided that the event happened if there's a match no less than in the five following frames too. This dataset tagging has a respectable 0.88 accuracy.

Marko Chavdar et al. [58] developed a robust system capable of traffic audio events detection in a real-life environment. The models are trained using log-mel spectrograms and MFCCs with chunks of different frame image lengths. Experiments with CNNs and Convolutional Variational Autoencoder (CVAE)s are made but the author concludes that CNNs based classifiers are best suited for the task.

Sound event detection gained huge popularity in recent years and many competitions are hold every year to find state of the art solutions for these type of tasks. In 2017 the second task of Detection and Classification of Acoustic Scenes and Events (DCASE) focused on rare sound event detection, this task focused on detecting the onset time of target events is still studied today with new approaches being proposed. The most accurate model was proposed by Hyungui Lim et al. [59] , which consists in a CRNN architecture composed by a 1D CNN and two layers of LSTM layers. The authors extract log-mel spectrograms from the audio and divide the spectrogram into frames with a timestep size that are fed to the network. Probability values of the event are calculated for each frame of the mel-spectrogram during the timestep and by sliding the prediction chunk the average of the probabilities sequence is calculated in order to combine them into a single probability sequence. The author finishes stating that bigger timesteps should be used for lengthier events (i.e. baby crying), likewise for transient sounds (i.e. gunshot) smaller timesteps yield better results. The second place of the competition also proposed a CRNN model with framewise classification stages but the features are fed to GRU layers opposed to the LSTM. Note that both the top 2 approaches relied on using ensemble method and post-processing for converting frame level predictions to event level predictions. Chieh-Chi Kao et al. attempted to deviate from this procedure by proposing a Region-based Convolutional Recurrent Neural Network (R-CRNN) [60]. The CRNN extracts features from the input spectrograms, uses ResNet pre-trained on ImageNet as the CNN with a bi-directional GRU layer. The region proposal network generates event proposals based on features and a final classifier uses non-maximum suppression to remove highly overlapped proposals. The model ends up performing slightly worse than the top 2 but still achieves good results considering the lack of ensemble methods especially when compared to other region-based methods that try to solve the same problem. Wang et al. brought forward a simple alternative recurrent model with multi-resolution feature extraction. Baumann et al. [61] showed a new approach to the same challenge with better results than the winners by developing the MWK-CRNN which is a CRNN with multi-width kernels, the authors extract log-mel spectrogram filterbank features and the first and second order temporal derivatives (delta and

delta delta) using all of them as an input feature vector.

Yanping Chen et al. [41] also used different datasets from DCASE challenges through the years to test a rare sound event detection system using deep learning and data augmentation. This system stands out in its data augmentation technique based on linear re-scaling and merging of multiple instances of the same event. The authors combine two neural networks classifiers with a fusion ensemble method: a Support Vector Machine (SVM) with radial basis function kernel for model fusion. The system consists of a CNN with spectrograms as inputs and a Feed Forward Neural Network (FNN) that takes MFCCs as inputs. The model is tested on multiple DCASEs datasets and gets better results than the compared competition baseline systems and mix-DA data augmentation technique.

Venkatesh et al. [62] proposed the You Only Hear Once (YOHO) algorithm, similar to You Only Look Once (YOLO) but with regard to audio segmentation and sound event detection. While most techniques treat sound event detection as a frame-based classification problem by focusing on splitting the audio signal in small frames and individually perform classification on those frames, the YOHO algorithm addresses sound event detection as a regression problem in a pure CNN architecture adapted from MobileNet, opposed to the state of the art CRNNs for audio event detection. This is achieved by having separate output neurons to detect the presence of an audio class and predict its onset and offset points. The input feature used is log-mel spectrogram. The authors evaluate the robustness of the algorithm on multiple datasets and use SpecAugment as a data augmentation the dataset is small. Compared to multiple state of the art CRNN and CNN architectures the YOHO algorithm generally has better F-scores and error rates and it's faster despite being larger.

Zhirong Ye et al. [63] proposed a one-dimensional Detection Transformer inspired by the detection transformer used for image object detection. This model gets rid of the frame-based predictions and generates event-level outputs directly. The architecture is composed by a ResNet-50 backbone CNN with mel-spectrograms as inputs. The CNN and the encoder of the transformer are used for feature extraction and the decoder generates event representations that ultimately are fed into FNNs for timestamp and class label generation. The transformer achives competitive performance when tested on the URBAN-SED dataset and the DCASE 2019 Task 4 dataset.

2.4 Metrics

To evaluate the performance of a given model it is important to have a good understating of which metrics to use. Sound event detection tasks where the data is strong labelled have two common approaches of evaluation: segment-based and event-based. The evaluation is done by comparing the reference annotation with the system output classifying events with binary indicators like True Positives (TPs), True Negatives (TNs), False Positives (FPs) and False Negatives (FNs).

Precision regards the proportion of predicted positive cases that are actually real positives [64].

Mathematically, this is stated as follows:

$$P = \frac{TP}{TP + FP} \tag{2.9}$$

Recall or True Positive Rate (TPR) regards the proportion of real positive cases that are correctly predicted positive [64]. Mathematically, this is stated as follows:

$$R = \frac{TP}{TP + FN} \tag{2.10}$$

F1-score or simply F-score, can be described as the harmonic mean between recall and precision [64], it is based on the total number of false positive, false negatives and true positives. Mathematically, this is stated as follows:

$$F = \frac{2TP}{2TP + FP + FN} \tag{2.11}$$

Error Rate is the total number of errors with respect to the total number of reference events N [50]. Mathematically, this is stated as follows:

$$ER = \frac{S + D + I}{N} \tag{2.12}$$

Where S stands for the single substitution error which translates in the joint occurrence of FPs and FNs, I stands for the FPs unaccounted in S and D stands for the FNs unaccounted in S.

Receiver Operating Characteristic (ROC) analysis plots the TPRs rate against the False Positive Rates (FPRs) with a curve [50]. The FPR is defined as:

$$FPR = \frac{FP}{FP + FN} \tag{2.13}$$

The Area Under the ROC Curve (AUC) is the two-dimensional area underneath the entire ROC curve, used to measure the model performance across all possible classification thresholds. It can be interpreted as the probability that the classifier ranks a random positive instance rather than a random negative one [50].

Segment-based evaluation is done comparing the true label to the system output by using segments with a determined length in time grill. The binary activity indicators are used to compare the reference annotation to the system output as seen in Fig. 23.

In event-based evaluation the temporal boundaries are attenuated. The *collar* acts like a room that allows some flexibility. An event instance is counted as a TP if it has the same label as the label reference and the temporal boundaries overlap with the temporal *collar*. It's up to the developer whether this condition has to be met in the beginning, end or both of the temporal boundaries. Figure 24 shows a 100 ms collar where both the onset and offset condition has to happen.



Figure 23: Segment-based evaluation: Comparison of binary indicators between the reference and output annotation, adapted from [50]

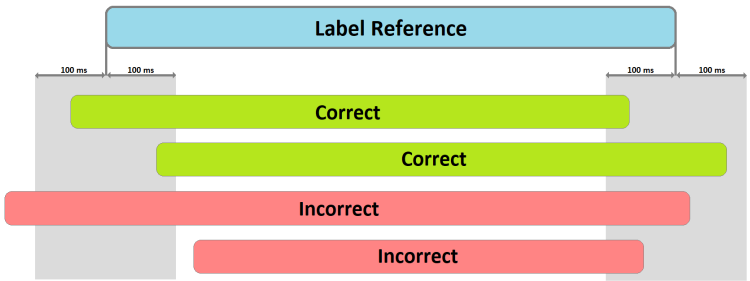


Figure 24: Event-based evaluation with a collar of 100 ms, adapted from [50]

Chapter 3

Development

In this section, we delve into the pratical work, discussing key aspects such as data collection, model selection, and the utilization of Transfer Learning and Fine Tuning techniques to build a robust car scratch detection algorithm. Deep Learning has rapidly emerged as the state-of-the-art solution for computer vision problems. The objective of this thesis is to document the development of a car 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 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 DNNs to address image detection and classification tasks in computer vision, as highlighted in the state-of-the-art research conducted during this thesis. Right away, it becomes apparent that one fundamental difficulty with the challenge will be the 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 thesis extensively documents a series of experiments conducted on audio and images, which were organized 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, utilizing the latest version of TensorFlow along with the Keras API, which provides comprehensive support for every step of the machine learning workflow. Scikit-learn was employed for predictive data analysis, Numpy for efficient array manipulation and linear algebra operations, and Matplotlib for creating interactive visualizations 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 network typically requires a substantial amount of data, however, collecting scratch data from cars with sensors can be a costly and time-consuming task, thus making the available data not sufficient 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 utilizing CNNs such as VGG16 or ResNet50, DL can be employed with relatively small amounts of data. These networks 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 they are not updated during training. Subsequently, the user can add a classifier for the new task and introduce additional new layers as needed. Figure 25 illustrates a general example of transfer learning.

Fine-tuning these networks 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 CNN networks for the problem at hand. Therefore, refining these pre-trained networks on large public datasets was a reasonable strategy to expedite the training process. It is important to note that the ImageNet dataset, on which these pre-trained networks were trained, does not contain spectrogram-like images. Consequently, the weights might not be optimized 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.

3.2 Data collection for scratch detection

The data collection phase was a collaborative effort involving multiple BOSCH facilities in Portugal and Germany. This thesis focus on performing scratch detection through the audio signal, therefore data acquisition had to respect this component by using a similar setup consisting of a microphone where audio is recorded. The audio clips used in this thesis were gathered using sensors positioned on the windshield near the rear-view mirror of various types of vehicles and under different conditions. While all

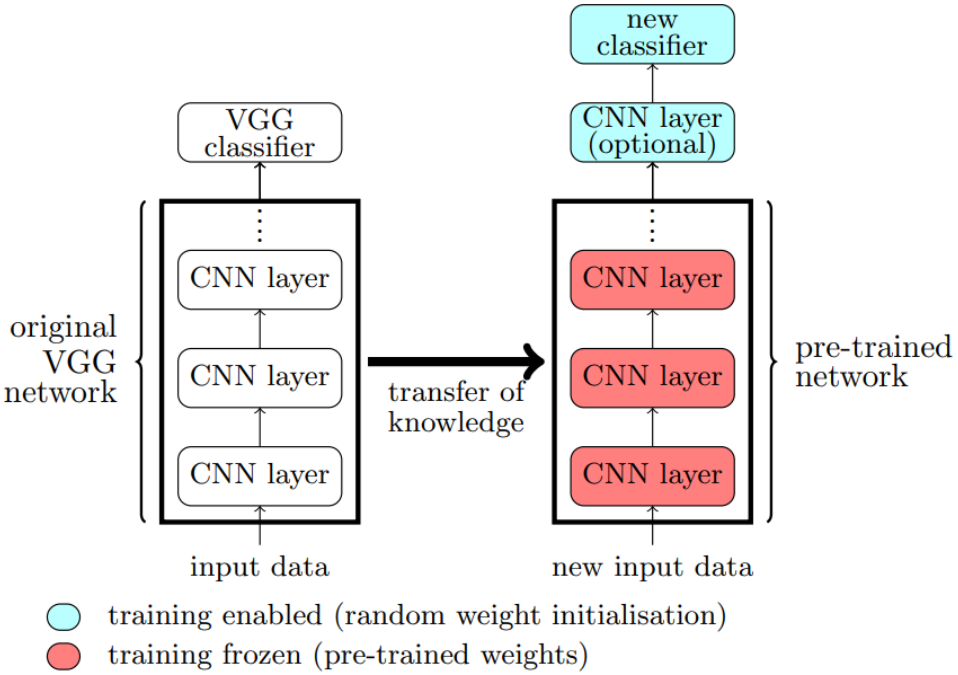


Figure 25: A general illustration of transfer learning/knowledge. The pre-trained NN on the left is utilised as a base (without the classifier) for the NN on the right.

scratch events were recorded with the vehicle in a stationary state, the inclusion of data from moving cars was essential, given 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 car is stationary.
- Talking: Recordings of conversations between passengers.
- Radio sound: Gathering audio with singing and unusual sounds that typically do not occur during normal passenger conversations.
- Car 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, impacts with objects and others.

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 audio-related events that do not generate damages to the vehicle and Table 3 enumerates the scratching events that make up our target class.

Description	Number of events
Vehicle hits Front Bumper	4
Object Impact Front Left Door	10
Object Impact Back Left Door	11
Object Impact Back Right Door	8
Object Impact Front Right Door	8
Object Impact Back Right Bumper	8
Door Opens against object Front Right	28
Object Impact Back Bumper	5
Door Opens against object Front Left	18
Object Impact Front Left Bumper	6
Object Impact Front Right Bumper	4
Object Impact Front Bumper	5
Object Impact Back Left Bumper	5
Door Opens against object Back Right	3
Door Opens against object Back Left	5
Vehicle hits Back Right Bumper	1
Object Impact Back Trunk	1

Table 1: Damage events

Description	Number of events
Door Open Front Left	14
Door Close Front Left	102
Door Open Back Left	13
Door Close Back Left	86
Door Open Trunk	7
Door Close Trunk	77
Door Open Back Right	13
Door Close Back Right	81
Door Open Front Right	27
Door Close Front Right	108
Door Close Hood	74

Description	Number of events
Object Impact Back Bumper	10
Object Impact Front Bumper	9
Object Impact Hood	2
Object Impact Back Left Bumper	6
Object Impact Back Right Bumper	3
Object Impact Front Right Bumper	2
Object Impact	57
Door Opens against object Front Right	8
Door Opens against object Back Right	3
Door Opens against object Front Left	4
Door Opens against object Back Left	5
Knock	208
Knock Back Right Door	34
Knock Front Right Door	43
Knock Hood	56
Knock Front Left Door	40
Knock Back Left Door	33
Knock Trunk	41
Roof Slap Center Inside	35
Roof Slap Back Left Outside	35
Roof Slap Front Left Outside	30
Roof Slap Front Right Outside	30
Roof Slap Back Right Outside	30
Roof Slap Center Outside	24
Roof Slap Back Left Inside	29
Roof Slap Front Left Inside	29
Roof Slap Front Right Inside	29
Roof Slap Back Right Inside	29
Windshield Slap Upper Left	26
Windshield Slap Bottom Left	26
Windshield Slap Bottom Right	26
Windshield Slap Upper Right	31
Windshield Slap Center	26
Dismounting Holder	28
Mounting Holder	27

Description	Number of events
Stomping	6
Opening Sun Visor Left	21
Closing Sun Visor Left	47
Open/Close Sun Visor	25
Closing Sun Visor Right	33
Open Sun Visor Right	8
Detaching Sun Visor Left	10
Fixing Sun Visor Left	10
Opening Makeup Mirror Right	8
Closing Makeup Mirror Right	8
Opening/Closing Makeup Mirror	49
Wiper	230
Close Mirror Folding Right	7
Open Mirror Folding Right	7
Mirror Folding Open/Close	24
Object Sliding Against Windshield Left	8
Object Sliding Against Windshield Center	8
Object Sliding Against Windshield Right	8
Ventilation	3
Car Wash	4
Object Impact Front Left Bumper	1
Door Open	4
Door Close	4
Object Impact Front Left Door	9
Object Impact Back Left Door	9
Object Impact Back Right Door	9
Object Impact Front Right Door	10
Detaching/Fixing Sun Visor	8
Opening Makeup Mirror Left	14
Closing Makeup Mirror Left	14
Close/Open Mirror Folding Left	17
Object Sliding Against Windshield	18
Roof Slap	103
Windshield Slap	66

Table 2: Non-damage events.

Description	Number of events
Scratch Back	54
Scrath Front	65
Scratch Door Back Left	14
Scratch Door Front Right	16
Scratch Door Back Right	24
Scratching	106
Scratch Door Front Left	25
Scratch Front Left Bumper	4

Table 3: Scratch events

Due to constraints during the data collection phase, capturing every conceivable type of event, such as "breaking windows,"was not possible. The primary focus of the collection efforts was on stationary data, as all scratch data was 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.

The table below provides details regarding the different cars used and the types of acquisitions:

Car Brand	Car Model	Acquisition
BMW	i3	Stationary
BMW	X1	Stationary
BMW	5	Stationary
Mini		Stationary
VW	Polo	Stationary
VW	Tiguan	Stationary
Smart		Stationary
Audi	A5	Stationary/Moving
Mercedes	A45	Stationary/Moving
Mercedes	GLA	Stationary

Table 4: Car models used in the data collection

A total of 513 data collections in audio format were considered for the development of this thesis. More details can be observed in table 5:

Car state	Total Collections	Total Collections time
Moving	9	50 min 45 sec
Stationary	504	4 h 21 min 1 sec

Table 5: Stationary and Moving data collection

Since data collection was conducted using various cars, diverse locations, and consequently different microphones, a crucial step was to establish a common ground through post-processing. This involved resampling all audio clips to a consistent sampling rate of 24,000 Hz 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 thesis's 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 data science techniques will be employed.

Type of event	Total Events	Total Events in hours
Non Scratch	2656	40 min 2 sec
Scratch	309	2 min 32 sec

Table 6: Scratch and non-scratch events distribution in data collection

3.3 Labeling

Data labeling 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 labeling should not be underestimated. Poorly labeled data can lead to very complex issues in the algorithm that are challenging to diagnose.

Initially, each audio clip underwent labeling with the onset and offset timestamps of scratch-related events. To ensure accurate labeling, the data labeling process was executed in two stages. The first labeling took place during the data collection phase, facilitated by internal BOSCH software that allowed real-time labeling using hotkeys. Subsequently, another piece of BOSCH software provided a user-friendly interface, enabling meticulous analysis of each clip. This interface allowed for signal visualization and playback, along with the verification of corresponding labels. Figure 26 displays an audio clip featuring a labelled section within BOSCH's internal software.

After labeling the collected data clips, spectrograms were generated using soft labels that denoted the percentage of a scratch event's presence within the respective window of analysis. For instance, consider a spectrogram corresponding to 0.5 seconds 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 seconds. This soft labeling 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 aproaches 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 thesis.

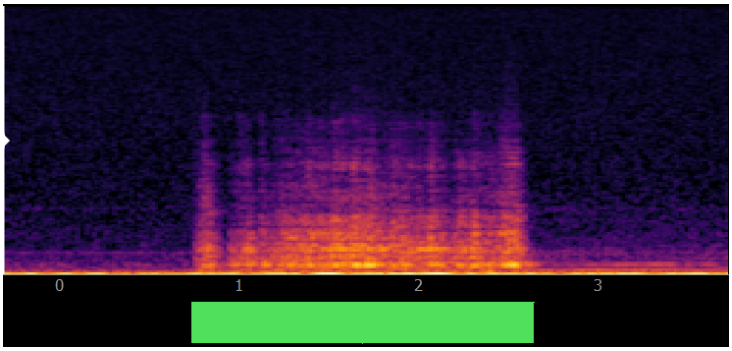


Figure 26: Example of label (in green) in internal BOSCH software

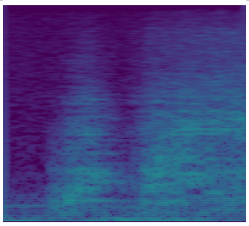
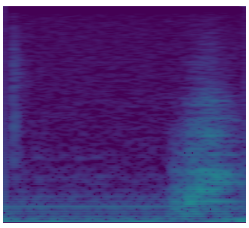
Spectrogram Image	Soft label	Hard label
	0.88	1
	0.28	0

Table 7: Difference between soft and hard labels for different spectrograms

3.4 Dataset specifications and Time-Frequency Representations

For the development of this thesis, three different datasets were created. Each dataset is split into 50%, 20%, and 30% for the training, validation, and testing sets, respectively. The splitting process involved a detailed listening analysis of individual audio sounds, ensuring a consistent representation of good/bad scratches and background events across all sets. This careful allocation aimed to maintain balance and relative consistency in the distribution of acoustic events, enhancing the reliability and fairness of the model evaluation.

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 scratch 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. Unfortunately, due to time constraints, testing every single idea across all datasets wasn't feasible. Therefore, more emphasis is placed on the best-performing strategies, aiming to consolidate and build upon the strengths of the preceding datasets as the project progresses forward.

These insights ultimately drove the necessary adjustments to tackle false predictions effectively. Notably, 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 utilizing limited physical memory. During training and validation, images are loaded in a random order to enhance generalization 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 figure 27. 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 STFT as it minimizes the effect of leakage by decreasing the signal amplitude near the boundaries of the samples [65].

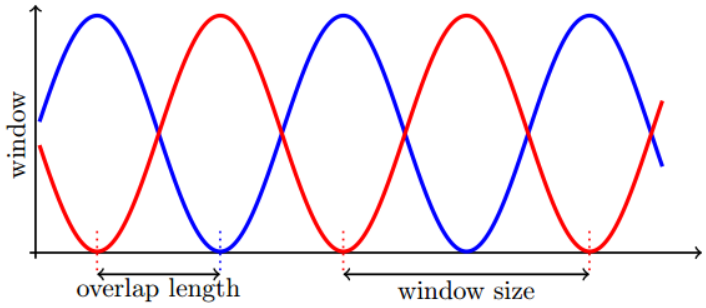


Figure 27: Window size and overlap length of the window function for STFT (adapted from [65])

The conversion process from audio to image represents one of the most critical aspects of this thesis. Given the half-second window analysis of audio signals, selecting the right features for network input from the previously studied transformations became imperative. In this study, possible signal transformations included Gammatoneagrams, Log Spectrograms, Log-Mel Spectrograms, and Chromagrams. Figure 28 illustrates an example of all these four transformations for the same scratch 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 networks. 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 normalization 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 falls

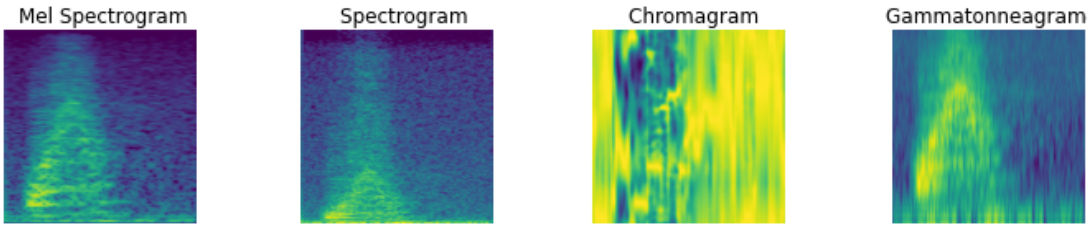


Figure 28: Different types of visual representations considered for the same scratch data

within a standardized range promoting stability and efficiency in the training process and contributing to improved model performance and generalization.

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

Dataset	Training	Validation	Testing
Dataset 1	913	323	533
Dataset 2	36927	12591	22185
Dataset 3	37612	12686	22318

Table 8: Layout of all datasets considered

3.4.1 Dataset 1 specifications

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 figure 29. 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 labeled scratch 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 below:

- Total number of data collection files: 91
- Total number of scratches related events: 166
- Total number of non-scratch related events: 0
- Total number of scratch related images: 361
- Total number of background related images: 1408

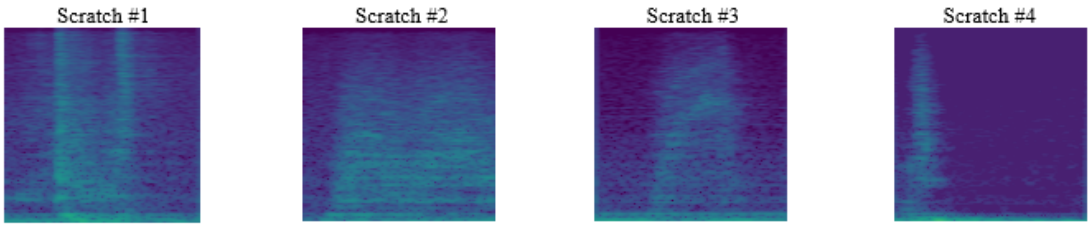


Figure 29: Example of scratches being pictured in log-Mel spectrograms

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 9, 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 gammatonneagram. 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.

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

Table 9: Distribution of images in train/test/validation sets featured in dataset 1

3.4.2 Dataset 2 specifications

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 car engines, passenger conversations, and live radio music, some examples of that kind of background events are illustrated in figure 30. A detailed breakdown of the dataset is provided below:

- Total number of data collection files: 465
- Total number of scratches related events: 249
- Total number of non-scratch related events: 2635
- Total number of scratch related images: 553

- Total number of non-scratch related images: 71150

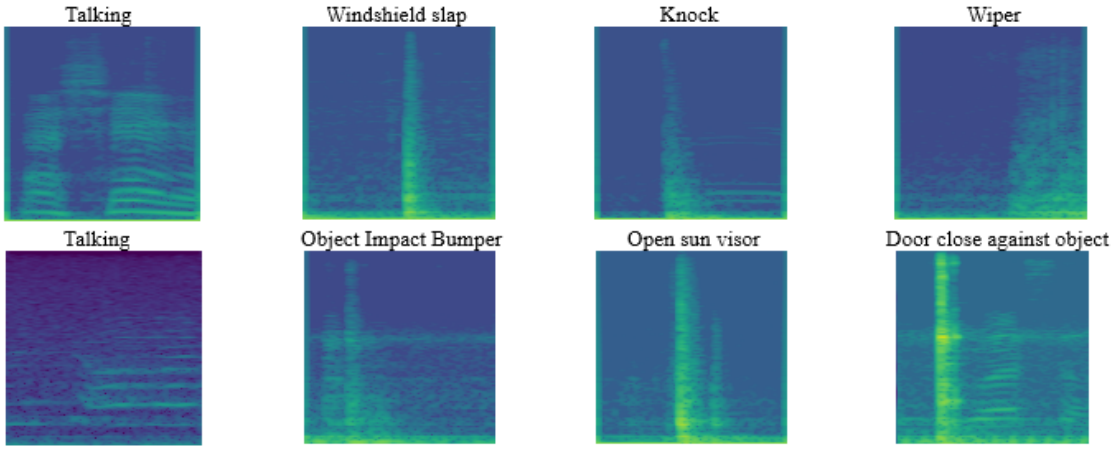


Figure 30: Example of different types of backgrounds events being pictured in log-Mel spectrograms

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 10.

Dataset	Scratch	Background
Training	339	36588
Validation	90	12501
Testing	124	22061

Table 10: Distribution of images in train/test/validation sets featured in dataset 2

3.4.3 Dataset 3 specifications

The third and final dataset in this project aims to replicate real-life conditions as closely as possible making it the chosen one for a possible live demo and ultimately, the product that will be sold to customers. As initial tests showed some difficulty with 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 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 scratch events within the spectrograms. Refer to Figure 31 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 11.

- Total number of data collection files: 513
- Total number of scratches related events: 309
- Total number of non-scratch related events: 2656
- Total number of scratch related images: 714
- Total number of non-scratch related images: 71902

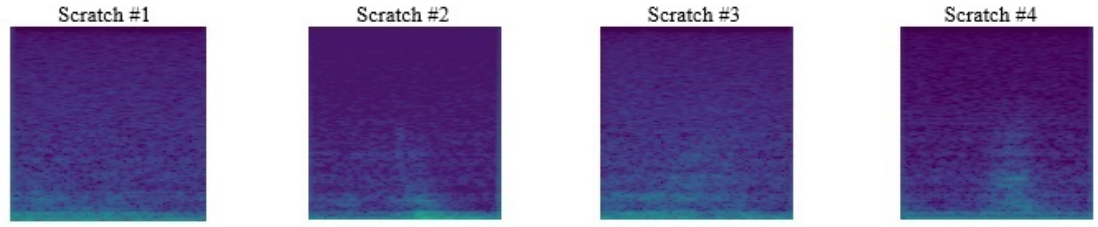


Figure 31: Example of more complex scratches being pictured in log-Mel spectrograms

Dataset	Scratch	Background
Training	438	37174
Validation	109	12577
Testing	167	22151

Table 11: Distribution of images in train/test/validation sets featured in dataset 3

3.5 Metrics

As the development of the algorithm progresses, it becomes increasingly important to establish robust metrics for evaluating and comparing its performance. In Chapter 2.4, we explored several potential metrics commonly used in audio detection and classification tasks. The problem 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: TN, TP, FN, and FP. These outcomes are often visualized in a two-by-two table known as a confusion matrix, illustrated in figure 32.

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, we have selected the Matthews Correlation Coefficient (MCC) as the primary metric for

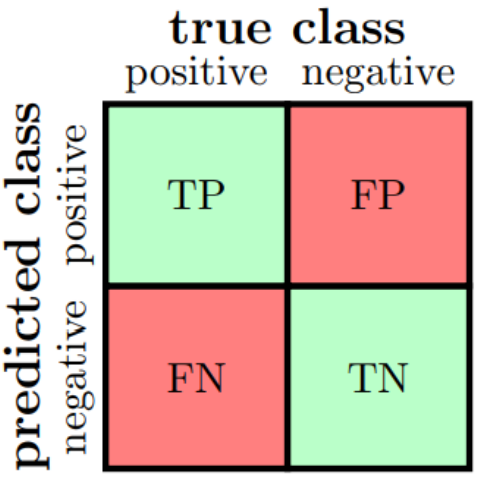


Figure 32: Confusion Matrix for Binary Classification

evaluating our models. MCC is a robust statistical measure that offers several advantages over metrics like F1 or accuracy.

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 [66]. The MCC formula presents as follows:

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{3.1}$$

The MCC scores range between -1 and +1, where:

- -1 : Missclassification
- 0 : Random prediction (coin tossing classifier)
- +1 : Perfect classification

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's crucial to recognize that using a default threshold of 0.5 may not be optimal for every problem, thus, the fact that this default treshold is commonly used in the literature it is important to do some threshold tuning and analysis.

To optimize the models, a threshold analysis was conducted. Figure 33 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 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 categorized as background.

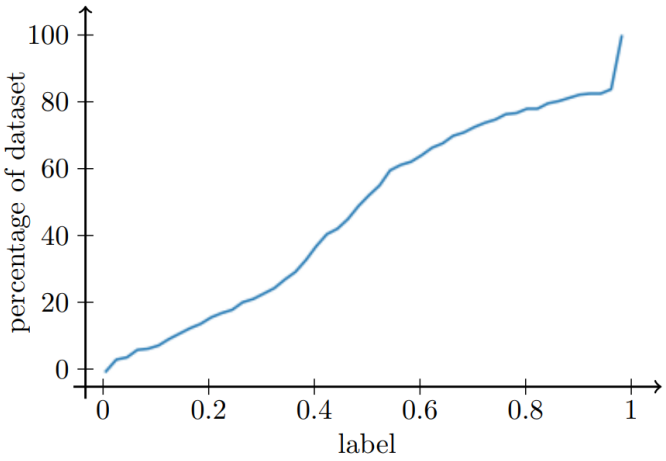


Figure 33: Accumulative percentage of scratches by label (only for images with label higher than 0).

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 our models' performance 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 deep learning models. Each real label of an image is compared to its predicted label, and based on a predefined threshold, it's categorized into one of the four confusion matrix categories. This metric offers a realistic assessment of the network's effectiveness, as it scrutinizes each image individually. This approach stands in contrast to the scratch-oriented metric, which will be discussed subsequently.

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 analyzed in a sliding window fashion with a fixed length, a single scratch 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's crucial to evaluate the algorithm's performance in real-life scenarios, where 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 scratch event.

Consequently, we developed a scratch-oriented classification, where a true positive signifies a correct prediction of a scratch 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 scratch 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, even if one or two of them fall below the detection threshold. This scratch-oriented metric provides a more holistic evaluation of the algorithm's performance in the context of scratch detection, ensuring that scratches are identified across all relevant images within a given event. For a visual representation of this concept, please refer to Figure 34.

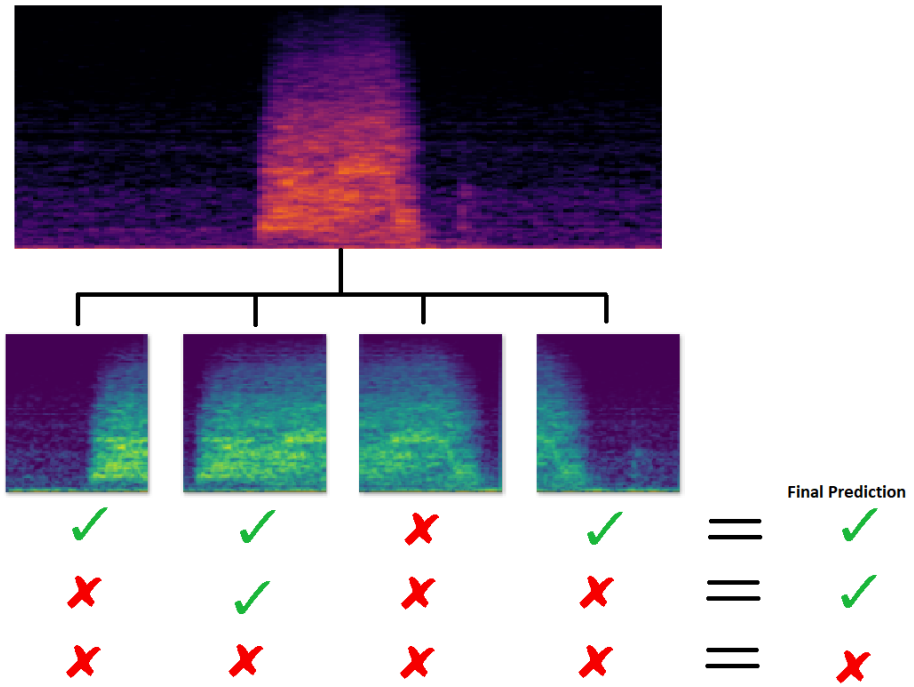


Figure 34: Scratch oriented classification

3.6 Exploring spectrogram mixing

In an attempt to merge features of multiple types of visual representations, a technique was developed 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 colors, 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 neural network would learn the fundamental changes between scratches and background data presented in the input data matrices. In the initial iteration of this spectrogram mixing approach, a combination of a non-mel-scaled spectrogram, a mel-scaled spectrogram, and a gammatoneagram was used. Examples of the resulting images can be observed in Figure 35.

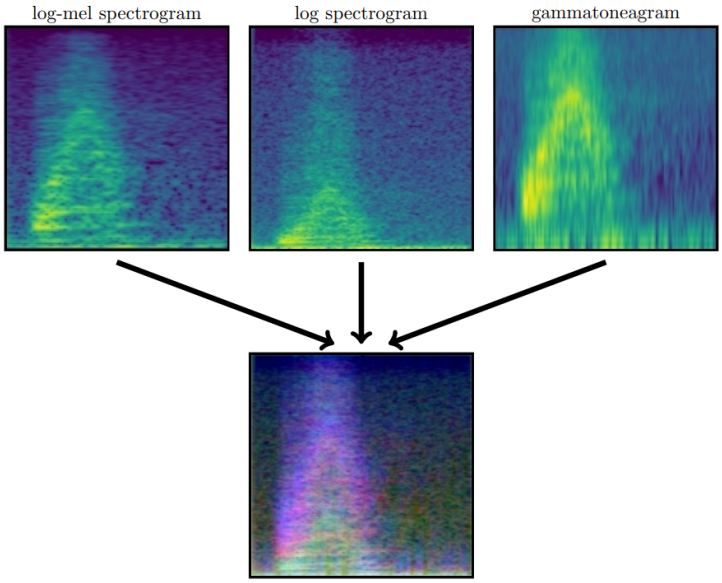


Figure 35: Example of log spectrogram, log-mel spectrogram and gammatoneagram mixing.

In the second iteration, the chromagram is introduced and combined with a log-mel-scaled spectrogram and the gammatoneagram. This approach aimed to further enrich the input data and enhance the network's ability to extract relevant features. Examples of the resulting images from this second iteration are depicted in Figure 36.

Spectrogram mixing represents an innovative approach to data representation, where diverse spectrogram types are integrated into a single image, potentially providing the neural network with a more comprehensive understanding of the audio data's intricate features, 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, the spectrogram images generated from the existing datasets were closely examined, and various relevant augmentation methods were rigorously tested on

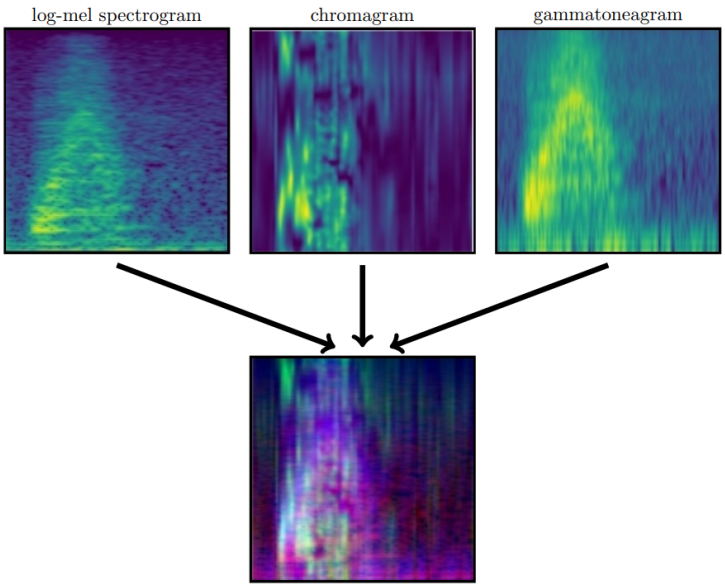


Figure 36: Example of log-mel spectrogram, chromagram, and gammatoneagram mixing.

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 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 emphasize 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 counterclockwise within a range of 0° to 30° degrees, altering the position of the spectrogram within the frame.

- 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 model performance [67].

Examples of these augmentations within the same spectrogram can be observed in image 37.

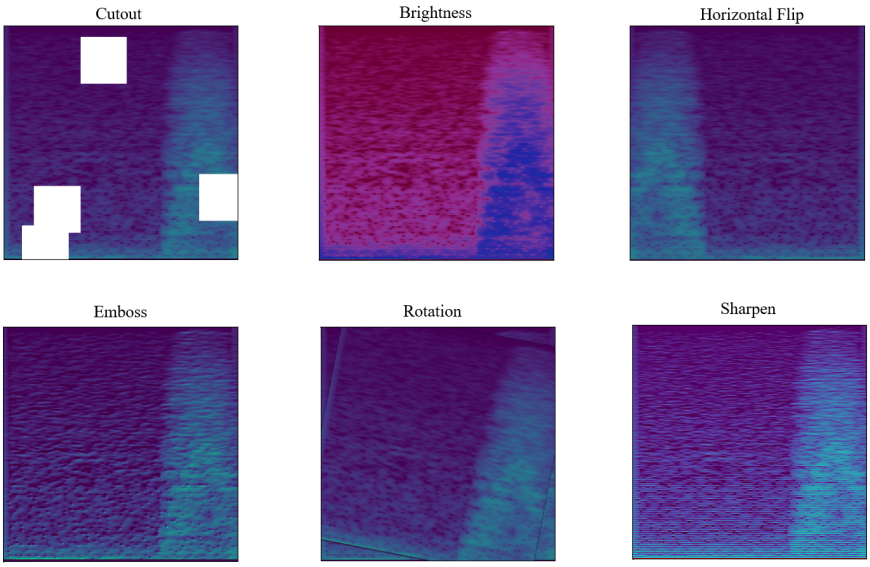


Figure 37: Multiple type of augmentations used in this work

3.8 Networks choice and parameters

Selecting the appropriate neural network architecture is a pivotal step in the development of the detection algorithm. Choosing the wrong network can significantly impact results and potentially lead to a slow, tiresome and frustrating development process. It is, therefore, imperative to deconstruct the problem, analyze it while considering available resources, and weigh key properties before making a decision:

- Input and type of problem: Since our 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 our 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 thesis is to implement the algorithm in a real-life scenario where scratch detection occurs in real-time, every 0.25 seconds relative to the 0.5 seconds of audio analysis with a 50% overlap. This underlines the importance of a fast inference speed.

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

VGG16 was used as a starting point for the experiments. All the development was conducted with this architecture. However, VGG16, while still performing at a high level, comes with certain limitations which led us to consider other networks especially in the implementation phase to a real life demo. Its sluggish training speed and over 130 million parameters make it less efficient compared to newer architectures. This inefficiency could present challenges when implementing the algorithm on a real-time audio recording device. Furthermore, VGG16 is an older network, prompting the exploration of more recent ones, unfortunately due to time constraints this was not possible. The final architecture of the VGG16 model used in this work is depicted in Figure 38.

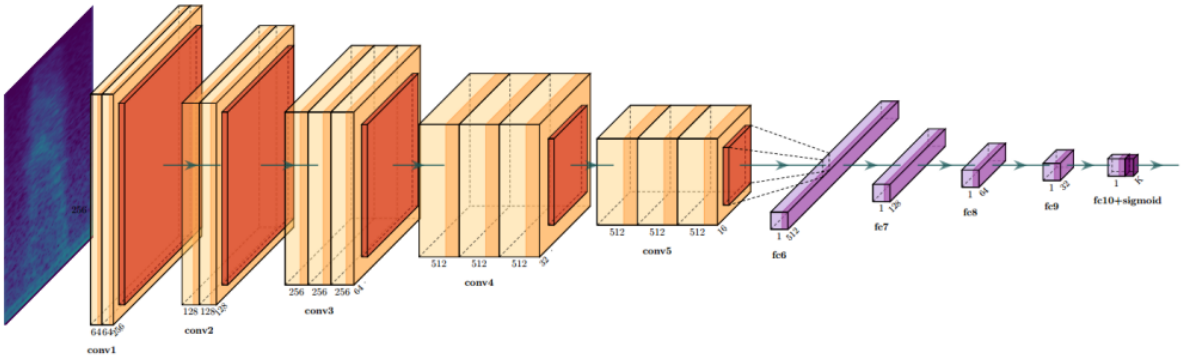


Figure 38: The architecture of the VGG16 model.

Beyond the choice of networks for experimentation, several fundamental principles and techniques related to deep learning 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, the sigmoid activation function was chosen because it effectively maps the network's output to a bounded range between 0 and 1. Given that our task involves binary classification (detecting scratches or not) with soft labels between 0 and 1 aswell, the sigmoid activation function is well-suited for producing probability-like values that indicate the likelihood of an 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 logcosh loss function is employed.

- Batch Size: Experimenting with different batch sizes can directly impact the training dynamics and efficiency of the NN employed for scratch detection. The batch size determines the number of training samples utilized in each iteration, influencing the model's weight updates. By systematically varying batch sizes, the objective is to understand how different-sized batches affect the convergence speed, generalization capabilities, and overall performance of the NN models.

Chapter 4

Results and Discussion

This chapter benchmarks and analyzes the different networks and datasets used in the dissertation according to the set of metrics defined to evaluate the performance of the models.

4.1 Dataset 1

The first dataset introduced in section 3.4.1 underwent a series of comprehensive experiments, laying the foundation for subsequent datasets. 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 usable features. These experiments primarily centered 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 treshold established in section 3.5.1.

4.1.1 Time-Frequency Representations

Our initial investigation revolved around determining the most suitable image representations for feeding the neural networks. Following the consensus in state-of-the-art research, which consistently advocated for log-mel spectrograms in sound event detection problems, we focused on these representations. This experiment involved computing various types of spectrograms using a sliding window of 0.5 seconds in length with a 50% overlap. Also at this stage, the experiments detailed in section 3.6, involving the use of a mix of multiple time-frequency representations in a single data-point, were conducted.

Representation	Image MCC	Scratch MCC
Log	0.85	0.79
Log Mel	**0.90**	**0.90**
Gamma	0.87	0.88
Chroma	0.72	0.73
LogMel+Mel+Gamma	0.85	0.83
LogMel+Chroma+Gamma	0.88	0.85

Table 12: Results for different types of spectrograms

4.1.2 Fourier Transform

Building upon our 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 summarized results can be found in table 13.

Fourier Transform	Image MCC	Scratch MCC
256X256	**0.90**	**0.90**
224X224	0.83	0.82
60x360	0.77	0.84

Table 13: Results for different types of mel-scaled spectrograms

4.1.3 Data Augmentation

To evaluate their performance, a range of augmentations was applied during training. The following table summarizes the outcomes, shedding light on the impact of the augmentations presented in 3.7.

Augmentation	Image MCC	Scratch MCC
No Augmentation	0.90	**0.90**
Horizontal Flip	**0.92**	0.88
Rotation	0.84	0.87
Cutout	0.90	0.88
Emboss	0.83	0.86
Sharpenning	0.86	0.88
Whitening	0.82	0.84

Table 14: Results for different types of mel-scaled spectrograms

4.1.4 Discussion

In the realm of potential options explored within Dataset 1, it became evident that log-mel spectrograms were the optimal choice. Furthermore, the decision to utilize squared images with a resolution of 256x256,

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 didn't really help in improving scratch detection, with the majority of the augmentations showing worse results compared to when no augmentation was applied.

As previously mentioned, the focus was always on reducing false positives. In the extensive analysis of the Dataset 1 FP, it became apparent that several missclassified spectrograms were shared across all the trained networks. Upon closer inspection of the corresponding audio data, emerged the discovery that many of these spectrograms captured events not initially labeled but 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's important to emphasize that Dataset 1 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 into the development of Dataset 2, we build upon the insights gained here to tackle more complex and real-life scenarios, therefore recognizing the significance of addressing the background sounds and unaccounted-for events that had inadvertently impacted the training process in Dataset 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.

4.2 Dataset 2

Building upon the knowledge gained from the results of the initial dataset in 4.1, Dataset 2 aimed to further refine our algorithm's performance in more complex and realistic scenarios. This second iteration of the dataset emphasized the inclusion of various potential background events that could potentially confound the network's ability to detect scratches, it is where most of development time was spent. Additionally, Dataset 2 introduced a substantial increase in the volume of data, with over 70,000 images, of which only 553 were associated with scratches (i.e., images with labels higher than 0). Due to the lessons learned in Dataset 1 we exclusively used log-mel spectrograms from this point on.

4.2.1 Data Augmentation & Batch Size

In this phase, we carried forward the augmentation techniques that yielded the best results in Dataset 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 summarized results can be found in table 15.

	No Augmentation		Horizontal Flip		Cutout		All Augmentations	
Batch Size	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	**0.90**	0.87	0.88	0.85	0.88
32	0.80	0.86	0.86	**0.90**	0.77	0.83	0.84	0.85
64	0.87	0.88	0.80	0.87	0.88	0.87	**0.90**	0.85
128	0.85	0.88	0.84	0.88	0.84	0.86	0.85	0.85

Table 15: Results for different augmentations and batch sizes on dataset 2

4.2.2 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, a critical evaluation was necessary to determine the optimal suitability of these pre-trained weights 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 initialized the model without pre-trained weights. This experiment aimed to determine if pre-trained weights might not align with our problem's performance requirements, particularly in relation to spectrograms. Once we'd already looked for what worked better in terms of augmentations and batch sizes, this experiment was done on the best results observed in table 15.

Network Specifications	S_MCC	I_MCC
Batch Size 32 - Horizontal Flip Only	**0.92**	**0.90**
Batch Size 64 - All Augmentations	0.90	0.89

Table 16: Results for different augmentations and batch sizes with no pre-trained weights

4.2.3 Best model results

Image 39 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's 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's accuracy. Analysing these deviations were fundamental in guiding decisions for the type of data that should be added next. One of approaches for this analysis is individually looking at false negatives and positives of the confusion matrices. In image 40, we provide a visual representation of the classification performance for Dataset 2 using both image-oriented and scratch-oriented metrics.

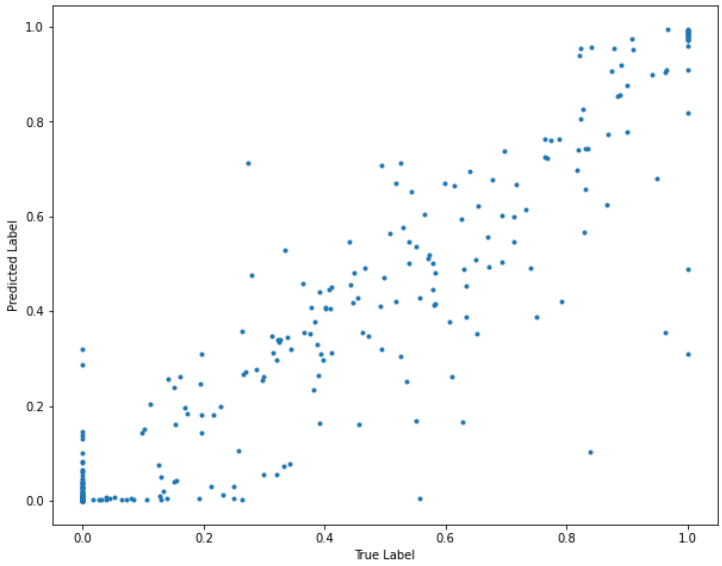


Figure 39: Prediction vs. Ground Truth Plot - Best Dataset 2 Model

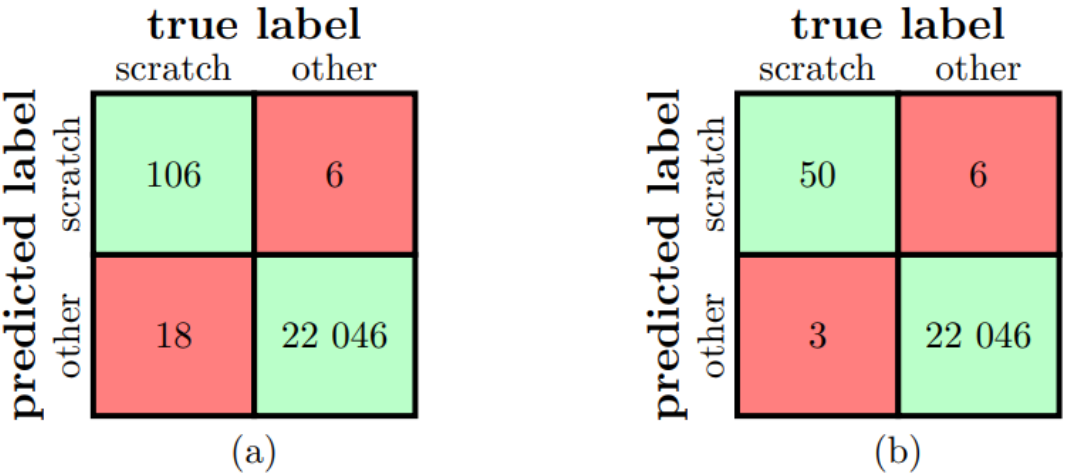


Figure 40: Confusion matrix for (a) image-oriented and (b) scratch-oriented metrics for the best dataset 2 model.

4.2.4 Discussion

The results obtained from Dataset 2 indicated that batch sizes of 32 and 64 yielded the most successful outcomes when combined with either horizontal flips or all augmentations applied simultaneously. Notably, training the base model from scratch without the pre-trained weights produced the best results, with both networks showing improvements in both the scratch and image oriented metric compared to their pre-trained counterparts. All these configurations that demonstrated superior scratch detection performance will be taken in consideration as the work proceeds to Dataset 3. While Dataset 2 witnessed a substantial expansion in terms of dataset size compared to Dataset 1, the primary objective of mitigating false positives was successfully accomplished. With this progress in mind, 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 model's sensitivity, such challenging cases were prioritized for inclusion in Dataset 3.

Dataset 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 our research.

4.3 Dataset 3

The third and final version of the dataset slightly expands the overall knowledge of different types of scratches. This is an attempt of generalizing the network into recognizing 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, 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.

	Pre-trained				No Pre-train			
	Horizontal Flip		All Augmentations		Horizontal Flip		All Augmentations	
Batch Size	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	**0.90**	**0.88**
64	0.75	0.83	0.77	0.82	0.84	0.82	0.82	0.81

Table 17: Results for different augmentations and batch sizes on dataset 3

4.3.1 Discussion

The results from Dataset 3, designed to emulate real-world conditions closely, underscore several important findings. Examining the outcomes presented in table 17, it is evident that the combination of all augmentation techniques detailed in section 3.7 consistently produced the most favorable results in both the image-oriented and scratch-oriented metrics. Unfortunately, the performance in dataset 3 did not meet the same standards as dataset 2, indicating that the network faced challenges in effectively detecting new low-amplitude scratches in this more complex real-world scenario. We can see that the distribution of prediction vs ground truth labels for the best model in dataset 3 in image 41 is worse than the equivalent dataset 2 represented in image 39. This goes in line with the increase of false positives and overall the number of false negatives in a individual image analysis only for a small decrease in false negatives in a scratch oriented analysis. This results are displayed in the confusion matrices in image 42, as usual, for the best model.

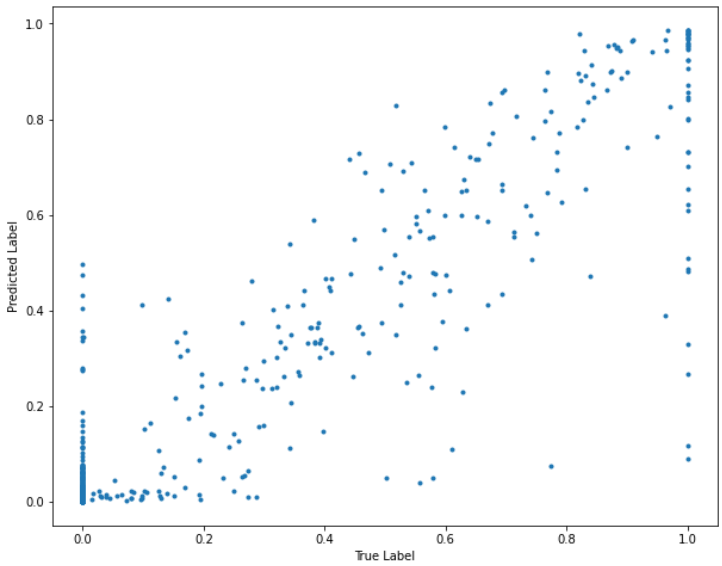


Figure 41: Prediction vs. Ground Truth Plot - Best Dataset 3 Model

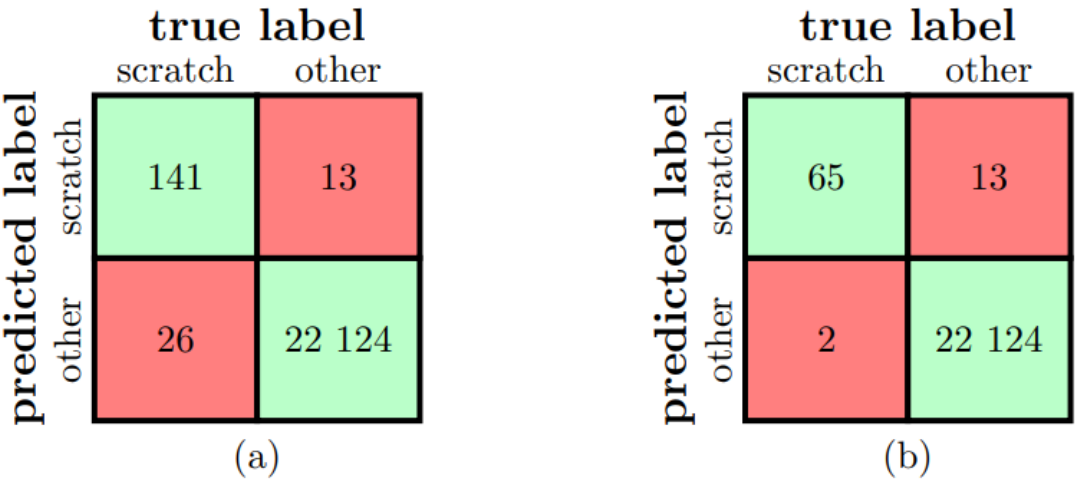


Figure 42: Confusion matrix for (a) image-oriented and (b) scratch-oriented metrics for the best dataset 3 model.

Chapter 5

Conclusion and Future work

This chapter presents a discussion of all the work done throughout the development of this dissertation, as well as the main conclusions and thoughts. Furthermore, we dive into possible future work by presenting additional approaches that were not explored in this thesis but that could be taken into consideration in order to improve the job done.

5.1 Conclusion

This research journey embarked with a clear set of objectives aimed at developing an advanced sound detection system, primed for the challenging task of identifying scratches within vehicular environments. 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.

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

In addition, let us now reflect on the work that has been done in relation to the objectives proposed in section 1.2:

- O1: The study conclusively demonstrated that scratch detection within the context of vehicular audio is possible and that neural networks 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 not only validated the feasibility of scratch detection but also facilitated the exploration of various data augmentation tecniques and spectrograms.

- O2: This objective was successfully achieved with Matthews Correlation Coefficient (MCC) scores of over 0.90 in the best models. Building upon the success of the first dataset, datasets 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.

- O3: Despite achieving success in both the first and second objectives, the work was not able to reach a predefined minimum threshold for the Matthews Correlation Coefficient (MCC) of 0.95 that was proposed in the objectives section and that was considered to be critical for a real-life implementation of the scratch detection algorithm for deployment in vehicles. This shortfall constitutes the primary setback in achieving the project objectives. The analysis of the results highlights a particular challenge encountered in the project: the difficulty neural networks face when accurately detecting scratches characterized by low amplitude and weak signals. In hindsight, it is evident that the efforts made to train the models to handle this type of scratches were not adequately supported by a sufficient quantity of relevant data within the dataset. This deficiency in representative examples of low-amplitude scratches limited the network's ability to successfully learn to identify them. To enhance the likelihood of reaching the target MCC value of 0.95, it is imperative to consider comprehensive data collection, emphasizing the inclusion of diverse samples that better reflect these subtle scratch types. Moreover, the continuous refinement of model architectures and the fine-tuning of hyperparameters 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.

To conclude, the usage of deep learning in in-vehicle scratch detection is still very much an open problem. Through the synergies of deep learning, diligent data collection and exploration, and comprehensive experimentation, this work has laid the foundation for future advancements in this field, the path ahead is rich with possibilities for further refinement and innovation.

5.2 Future Work

While the developed material can be seen as promising, there is still work to be done. The field of audio-based car scratch detection presents numerous challenges and 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, cross-validation and active learning could be explored to address this imbalance and enhance model performance.

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.

Considering the ever-evolving landscape of deep learning, exploring new model architectures holds significant promise. Consideration of more advanced model architectures, such as EfficientNet [68] and Resnet [19] 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 car scratch detection considering their promise at capturing intricate patterns in data.

Bibliographic References

[1] N. Aloysius and M. Geetha. "A review on deep convolutional neural networks". In: *6th International Conference on Communication and Signal Processing (ICCSP 2017)*. DOI 10.1109/ICCSP.2017.8286426. IEEE, 2017, pp. 588–592 (cit. on pp. 4, 8).

[2] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization". In: DOI 10.48550/arXiv.1412.6980. Dec. 2014 (cit. on p. 5).

[3] W. S. McCulloch and W. Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The Bulletin of Mathematical Biophysics* 5.4 (1943). DOI 10.1007/BF02478259, pp. 115–133 (cit. on p. 5).

[4] F. Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain". In: *Neurocomputing, Volume 1: Foundations of Research*. DOI 10.7551/mitpress/4943.003.0010. The MIT Press, 1988 (cit. on p. 5).

[5] M. Minsky and S. A. Papert. "Perceptrons". In: *Neurocomputing, Volume 1: Foundations of Research*. DOI 10.7551/mitpress/4943.003.0015. The MIT Press, Apr. 1988 (cit. on p. 5).

[6] J. J. Hopfield. "Neural networks and physical systems with emergent collective computational abilities". In: *National Academy of Sciences* 79.8 (1982). DOI 10.1073/pnas.79.8.2554, pp. 2554–2558 (cit. on p. 5).

[7] G. E. Hinton, S. Osindero, and Y.-W. Teh. "A Fast Learning Algorithm for Deep Belief Nets". In: *Neural Comput.* 18.7 (July 2006). DOI 10.1162/neco.2006.18.7.1527, pp. 1527–1554 (cit. on p. 5).

[8] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning". In: *Nature* 521 (2015). DOI 10.1038/nature14539, pp. 436–444 (cit. on pp. 5, 7).

[9] X. Du et al. "Overview of deep learning". In: *31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*. DOI 10.1109/YAC.2016.7804882. 2016, pp. 159–164 (cit. on p. 5).

[10] N. Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014). DOI 10.5555/2627435.2670313, pp. 1929–1958 (cit. on p. 6).

[11] L. Perez and J. Wang. "The Effectiveness of Data Augmentation in Image Classification using Deep Learning". In: *ArXiv* (2017). DOI 10.48550/arXiv.1712.04621 (cit. on pp. 6, 7).

[12] L. Shangzheng. “A Traffic Sign Image Recognition and Classification Approach Based on Convolutional Neural Network”. In: *11th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA 2019)*. DOI 10.1109/ICMTMA.2019.00096. 2019, pp. 408–411 (cit. on p. 7).

[13] J. Perry and A. Fernandez. “MinENet: A Dilated CNN for Semantic Segmentation of Eye Features”. In: *17th IEEE/CVF International Conference on Computer Vision Workshop (ICCVW 2019)*. DOI 10.1109/ICCVW.2019.00453. IEEE, 2019, pp. 3671–3676 (cit. on p. 7).

[14] D. Qu et al. “An Automatic System for Smile Recognition Based on CNN and Face Detection”. In: *IEEE International Conference on Robotics and Biomimetics (ROBIO 2018)*. DOI 10.1109/ROBIO.2018.8665310. IEEE, 2018, pp. 243–247 (cit. on p. 7).

[15] Y. Lecun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998). DOI 10.1109/5.726791, pp. 2278–2324 (cit. on p. 8).

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Communications of the ACM* 60.6 (2017). DOI 10.1145/3065386, pp. 84–90 (cit. on p. 9).

[17] Huimin Li et al. “A high performance FPGA-based accelerator for large-scale convolutional neural networks”. In: *26th International Conference on Field Programmable Logic and Applications (FPL 2016)*. DOI 10.1109/FPL.2016.7577308. IEEE, 2016, pp. 1–9 (cit. on p. 9).

[18] K. Simonyan and A. Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *3rd International Conference on Learning Representations (ICLR 2015)*. URL http://arxiv.org/abs/1409.155 2015 (cit. on p. 9).

[19] K. He et al. “Deep Residual Learning for Image Recognition”. In: *29th IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*. DOI 10.1109/CVPR.2016.90. June 2016, pp. 770–778 (cit. on pp. 10, 59).

[20] C. Szegedy et al. “Going deeper with convolutions”. In: *28th IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2015)*. DOI 10.1109/CVPR.2015.7298594. 2015, pp. 1–9 (cit. on p. 10).

[21] V. Bawa. *Basic Architecture of RNN and LSTM*. url: `https://pydeeplearning.weebly.com/blog/basic-architecture-of-rnn-and-lstm` (cit. on p. 11).

[22] J. Cheng, L. Dong, and M. Lapata. “Long Short-Term Memory-Networks for Machine Reading”. In: *2016 Conference on Empirical Methods in Natural Language Processing*. DOI 10.18653/v1/D16-1053. Nov. 2016, pp. 551–561 (cit. on pp. 11–13).

[23] HochreiterSepp and SchmidhuberJürgen. “Long Short-Term Memory”. In: *Neural Computation* (1997). DOI 10.1162/neco.1997.9.8.1735 (cit. on p. 11).

[24] J. Chung et al. “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling”. In: *NIPS 2014 Workshop on Deep Learning, December 2014*. DOI 10.48550/arXiv.1412.3555. 2014 (cit. on p. 11).

[25] G. E. Hinton and R. R. Salakhutdinov. “Reducing the Dimensionality of Data with Neural Networks”. In: *Science* 313.5786 (2006). DOI 10.1126/science.1127647, pp. 504–507 (cit. on pp. 11, 12).

[26] Z. Cheng et al. “Deep Convolutional AutoEncoder-based Lossy Image Compression”. In: *Picture Coding Symposium*. 2018, pp. 253–257. doi: 10.1109/PCS.2018.8456308 (cit. on p. 12).

[27] J. Liang and K. Kelly. “Training Stacked Denoising Autoencoders for Representation Learning”. In: *ArXiv* (2021). DOI 10.48550/arXiv.2102.08012 (cit. on p. 12).

[28] P. J. Pereira et al. “Using Deep Autoencoders for In-vehicle Audio Anomaly Detection”. In: *Procedia Computer Science* 192 (2021). DOI 10.1016/j.procs.2021.08.031, pp. 298–307 (cit. on p. 12).

[29] Q. Meng et al. “Relational autoencoder for feature extraction”. In: DOI 10.1109/IJCNN.2017.7965877. IEEE, 2017, pp. 364–371 (cit. on p. 12).

[30] D. Bahdanau, K. Cho, and Y. Bengio. “Neural Machine Translation by Jointly Learning to Align and Translate”. In: *3rd International Conference on Learning Representations*. 2015. doi: 10.48550/arXiv.1409.0473 (cit. on pp. 12, 13).

[31] A. Vaswani et al. “Attention is All You Need”. In: *31st International Conference on Neural Information Processing Systems*. NIPS’17. 2017, pp. 6000–6010 (cit. on pp. 13, 14).

[32] A. Dosovitskiy et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: (2021). url: https://openreview.net/forum?id=YicbFdNTTy (cit. on p. 13).

[33] J. Bi, Z. Zhu, and Q. Meng. “Transformer in Computer Vision”. In: *IEEE International Conference on Computer Science, Electronic Information Engineering and Intelligent Control Technology (CEI 2021)*. 2021, pp. 178–188. doi: 10.1109/CEI52496.2021.9574462 (cit. on p. 13).

[34] W. Dai et al. “Very deep convolutional neural networks for raw waveforms”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing*. 2017, pp. 421–425. doi: 10.1109/ICASSP.2017.7952190 (cit. on p. 15).

[35] “End-to-end environmental sound classification using a 1D convolutional neural network”. In: *Expert Systems with Applications* 136 (2019). DOI 10.1016/j.eswa.2019.06.040, pp. 252–263 (cit. on p. 15).

[36] R. N. Bracewell. *The Fourier transform and its applications*. 1965 (cit. on p. 15).

[37] L. Hertel, H. Phan, and A. Mertins. “Comparing time and frequency domain for audio event recognition using deep learning”. In: *International Joint Conference on Neural Networks*. 2016, pp. 3407–3411. doi: 10.1109/IJCNN.2016.7727635 (cit. on p. 15).

[38] M. Huzaifah. "Comparison of Time-Frequency Representations for Environmental Sound Classification using Convolutional Neural Networks". In: *ArXiv* (2017). DOI 10.48550/arXiv.1706.07156 (cit. on pp. 16, 17).

[39] A. M. Badshah et al. "Speech Emotion Recognition from Spectrograms with Deep Convolutional Neural Network". In: DOI 10.1109/PlatCon.2017.7883728. IEEE, 2017, pp. 1–5 (cit. on p. 17).

[40] J. Hall, W. O'Quinn, and R. J. Haddad. "An Efficient Visual-Based Method for Classifying Instrumental Audio using Deep Learning". In: DOI 10.1109/SoutheastCon42311.2019.9020571. IEEE, 2019, pp. 1–4 (cit. on p. 17).

[41] Y. Chen and H. Jin. "Rare Sound Event Detection Using Deep Learning and Data Augmentation". In: *Interspeech*. DOI 10.21437/Interspeech.2019-1985. ISCA, 2019, pp. 619–623 (cit. on pp. 17, 25).

[42] *Sound visualization, Signal Processing*. 2019. url: https://maelfabien.github.io/machinelearning/Speech10/#linear-frequency-power-spectrogram (cit. on p. 17).

[43] *Librosa - Feature extraction, Spectral features*. url: https://librosa.org/doc/main/feature.html#spectral-features (cit. on p. 19).

[44] Ali Shoeb and Gari Clifford. "Chapter 16 - Wavelets; Multiscale Activity in Physiological Signals". In: *Biomedical Signal and Image Processing Spring 2005* (cit. on pp. 18, 19).

[45] K. Ajith, R. Menaka, and S. S. Kumar. "EEG based mental state analysis". In: *Journal of Physics: Conference Series* 1911.1 (2021). DOI 10.1088/1742-6596/1911/1/012014, p. 012014 (cit. on p. 20).

[46] J. C. Brown. "Calculation of a constant Q spectral transform". In: *The Journal of the Acoustical Society of America* 89.1 (1991). DOI 10.1121/1.400476, pp. 425–434 (cit. on p. 20).

[47] C. Mateo and J. A. Talavera. "Bridging the gap between the short-time Fourier transform (STFT), wavelets, the constant-Q transform and multi-resolution STFT". In: *Signal, Image and Video Processing* 14.8 (2020), pp. 1535–1543. doi: 10.1007/s11760-020-01701-8 (cit. on p. 20).

[48] P. Singh, G. Saha, and M. Sahidullah. "Non-linear frequency warping using constant-Q transformation for speech emotion recognition". In: *2021 International Conference on Computer Communication and Informatics (ICCCI)*. 2021, pp. 1–6. doi: 10.1109/ICCCI50826.2021.9402569 (cit. on p. 20).

[49] *Librosa - constant-Q transform of an audio signal*. url: https://librosa.org/doc/main/generated/librosa.cqt.html (cit. on p. 21).

[50] A. Mesaros et al. "Sound Event Detection: A tutorial". In: *IEEE Signal Processing Magazine* 38.5 (2021), pp. 67–83. doi: 10.1109/MSP.2021.3090678 (cit. on pp. 20, 26, 27).

[51] R. Patterson et al. “An efficient auditory filterbank based on the gammatone function”. In: (1987) (cit. on p. 21).

[52] A. F. Pour, M. Asgari, and M. R. Hasanabadi. “Gammatonegram based speaker identification”. In: DOI 10.1109/ICCKE.2014.6993383. IEEE, 2014, pp. 52–55 (cit. on p. 21).

[53] G. H. Wakefield and G. H. Wakefield. “Mathematical representation of joint time-chroma distributions”. In: *Advanced Signal Processing Algorithms, Architectures, and Implementations IX* 3807 (1999). DOI 10.1117/12.367679, pp. 637–645 (cit. on pp. 21, 22).

[54] *Librosa - chromagram*. url: https://librosa.org/doc/main/generated/librosa.feature.chroma_stft.html (cit. on p. 22).

[55] L. Muda, M. Begam, and I. Elamvazuthi. “Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques”. In: *CoRR* abs/1003.4083 (2010). DOI 10.48550/arXiv.1003.4083 (cit. on p. 22).

[56] R. Müller. et al. “Acoustic Anomaly Detection for Machine Sounds based on Image Transfer Learning”. In: *13th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART*. SciTePress, 2021, pp. 49–56. doi: 10.5220/0010185800490056 (cit. on p. 23).

[57] S. A. Romanov et al. “Development of an Non-Speech Audio Event Detection System”. In: *2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*. 2020, pp. 1421–1423. doi: 10.1109/EIConRus49466.2020.9039115 (cit. on p. 24).

[58] M. Chavdar et al. “Towards a system for automatic traffic sound event detection”. In: *2020 28th Telecommunications Forum (TELFOR)*. 2020, pp. 1–4. doi: 10.1109/TELFOR51502.2020.9306592 (cit. on p. 24).

[59] H. Lim, J.-S. Park, and Y. Han. “Rare Sound Event Detection Using 1D Convolutional Recurrent Neural Networks.” In: *DCASE*. 2017, pp. 80–84 (cit. on p. 24).

[60] C.-C. Kao et al. “R-CRNN: Region-based Convolutional Recurrent Neural Network for Audio Event Detection”. In: *Interspeech 2018*. DOI 10.21437/Interspeech.2018-2323. ISCA, 2018, pp. 1358–1362 (cit. on p. 24).

[61] J. Baumann et al. “A New DCASE 2017 Rare Sound Event Detection Benchmark Under Equal Training Data: CRNN With Multi-Width Kernels”. In: DOI 10.1109/ICASSP39728.2021.9414254. IEEE, 2021, pp. 865–869 (cit. on p. 24).

[62] S. Venkatesh, D. Moffat, and E. Miranda. “You Only Hear Once: A YOLO-like Algorithm for Audio Segmentation and Sound Event Detection”. In: *Applied Sciences* 12 (2022). DOI 10.3390/app12073293, p. 3293 (cit. on p. 25).

[63] Z. Ye et al. “Sound Event Detection Transformer: An Event-based End-to-End Model for Sound Event Detection”. In: *CoRR* abs/2110.02011 (2021). DOI 10.48550/arXiv.2110.02011 (cit. on p. 25).

[64] D. Powers and Ailab. "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation". In: *J. Mach. Learn. Technol* 2 (2011). DOI 10.9735/2229-3981, pp. 2229–3981 (cit. on pp. 25, 26).

[65] M. Trethewey. "Window and overlap processing effects on power estimates from spectra". In: *Mechanical Systems and Signal Processing* 14.2 (2000). DOI 10.1006/mssp.1999.1274, pp. 267–278 (cit. on p. 37).

[66] D. Chicco and G. Jurman. "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation". In: *BMC Genomics* 21.1 (2020). DOI 10.1186/s12864-019-6413-7 (cit. on p. 42).

[67] T. Devries and G. W. Taylor. "Improved Regularization of Convolutional Neural Networks with Cutout". In: *CoRR* abs/1708.04552 (2017). doi: 10.48550/arXiv.1708.04552 (cit. on p. 47).

[68] M. Tan and Q. V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *36th International Conference on Machine Learning (ICML 2019)*. 2019, pp. 6105–6114 (cit. on p. 59).