FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Automotive Interior Sensing: Human Interaction Recognition

Maria Carolina Silva Teixeira Pinto

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor @ FEUP: Prof. Jaime Cardoso

Supervisor @ Bosch: Joaquim Fonseca

January 25, 2021

Resumo

Com a revolução dos veículos autónomos partilhados que se antevém nos próximos anos, deixa de haver um condutor responsável pelo bem-estar dos passageiros tal como de preservar o interior do carro. Sendo assim, é crucial existir um sistema de monitorização que seja capaz de detetar diferentes interações entre os passageiros, como por exemplo, de forma a ser possível identificar potenciais situações de escalamento de interações ou mesmo violência.

Esta dissertação recorre a técnicas de *Computer Vision* e de *Deep Learning* para resolver o problema de reconhecimento de ações e interações, no contexto de veículos autónomos partilhados. Inicialmente, duas arquiteturas diferentes foram testadas para a tarefa mais simples de deteção de violência: a I3D (*Inflated Inception network*) e a LRCN (*Long-term Recurrent convolutional Network*). Para além disso, os resultados obtidos com estas arquiteturas foram também comparados com os resultados de uma *framework* de deteção de anomalias. A melhor arquitetura foi então escolhida para a tarefa mais complexa de deteção de interações. Estas redes foram treinadas com um *dataset* interno da Bosch que inclui várias interações diferentes entre passageiros dentro de um veículo. Adicionalmente, foram feitas várias experiências para melhorar a robustez do algoritmo, utilizando informação de áudio adicional para a tarefa semelhante de reconhecimento de emoções. Por fim, foi também investigado o uso de *motion features* com o objetivo de melhorar a capacidade de generalização do modelo.

A *framework* resultante foi avaliada recorrendo a métricas tipicamente usadas para problemas de classificação e os resultados indicam que esta é uma opção viável para o reconhecimento de interações entre passageiros num veículo. Por outro lado, foi feita uma avaliação mais detalhada em relação à *performance* da rede recorrendo a uma toolbox de avaliação fornecida pela Bosch.

Palavras-chave: Veículos Autónomos Partilhados, Reconhecimento de Interações, Deep learning.

Abstract

With the emergence of Shared Autonomous Vehicles (SAVs) there is no longer a driver in the car responsible for ensuring the safety of the passengers as well as preserving the car's interior. Therefore, an interior analysis becomes essential for the car to have a "sense" of what is going on. Recognizing interactions between passengers is key to understanding what is happening inside the vehicle and, thus, a system capable of understanding different interactions as well as escalation levels should be developed.

This dissertation addresses the problem of in-vehicle human interaction recognition. In particular, it focuses on the use of computer vision and deep learning techniques to solve the problem of action recognition. Initially, two different network architectures were selected for the simpler task of violence detection: the I3D (Inflated Inception network) and a LRCN (Long-term Recurrent Convolutional Network). Furthermore, their results were benchmarked against an anomaly detection framework. The best performing network was then used for the more complex task of interaction recognition. All these networks were trained using one of Bosch's internal datasets that include several different types of interactions between passengers inside a vehicle. Additionally, several experiments were performed regarding the use of audio information for a more robust classification for the similar task of emotion recognition. Finally, the use of motion features was investigated in an attempt to improve the model's ability to generalize.

The resulting framework was evaluated using the most common metrics for imbalanced classification problems and results indicate that it is a viable option for in-vehicle interaction recognition. Moreover, an evaluation toolbox provided by Bosch is also used to provide a more detailed assessment regarding network performance for the simpler task of violence detection.

Keywords: Shared Autonomous Vehicles, Interaction recognition, In-vehicle, Deep learning.

Acknowledgements

I would like to thank my supervisor Prof. Jaime Cardoso from the the Faculty of Engineering of University of Porto and Eng. Joaquim Fonseca from the Bosch Car Multimedia Group. They provided continuous support and guidance that were essential throughout this dissertation.

I would also like to thank INESC TEC for providing me with the opportunity of working with them on the EmotiW competition. Furthermore, I would like to express my gratitude to Bosch Car Multimedia for providing a good work environment and to the members of the "In-Vehicle Sensing" team for their availability and help in the integration process.

Finally, I would like to express my deep appreciation for my family for the motivation and support offered throughout this dissertation.

Carolina Pinto

This work is supported by: European Structural and Investment Funds in the FEDER component, through the Operational Competitiveness and Internationalization Programme (COMPETE 2020) [Project nº 039334; Funding Reference: POCI-01-0247-FEDER-039334].

"You should be glad that bridge fell down.
I was planning to build thirteen more to that same design"

Isambard Kingdom Brunel

Contents

1 Introduction **1**
- 1.1 Contributions . . . 2
- 1.2 Document structure . . . 2

2 Related Work **3**
- 2.1 Traditional approaches . . . 4
  - 2.1.1 Handcrafted features . . . 4
- 2.2 Machine learning and deep learning . . . 6
  - 2.2.1 Datasets . . . 7
  - 2.2.2 Artificial Neural Networks (ANNs) . . . 8
  - 2.2.3 Convolutional Neural Networks (CNNs) . . . 11
  - 2.2.4 Recurrent Neural Networks (RNNs) . . . 12
  - 2.2.5 The classification task . . . 13
- 2.3 State-of-the-art approaches . . . 17
  - 2.3.1 Standard action recognition datasets . . . 17
  - 2.3.2 Deep-learning approaches . . . 18

3 Problem Description **25**
- 3.1 Problem definition . . . 25
- 3.2 Proposed solution . . . 26
- 3.3 Alternative solutions . . . 26

4 Emotion Recognition **27**
- 4.1 Introduction . . . 28
- 4.2 Related Work . . . 28
- 4.3 Implementation . . . 29
  - 4.3.1 Dataset . . . 29
  - 4.3.2 Baseline . . . 30
  - 4.3.3 Video-based emotion recognition module . . . 30
  - 4.3.4 Audio-based emotion recognition module . . . 30
  - 4.3.5 Score-level ensemble . . . 32
  - 4.3.6 Training . . . 32
  - 4.3.7 Evaluation . . . 33
- 4.4 Results . . . 33
- 4.5 Conclusion . . . 35

5 Interaction recognition **37**
5.1 Internal datasets . . . 38
5.1.1 Data splits . . . 38
5.2 Framework implementation . . . 39
5.2.1 Data preparation . . . 40
5.3 Violence detection . . . 40
5.3.1 Network architecture . . . 41
5.3.2 Class imbalance . . . 44
5.4 Multi-class classification . . . 45
5.5 Evaluation . . . 45
5.5.1 Evaluation toolbox . . . 45
5.5.2 Visualization tool . . . 47
5.6 Model optimization . . . 47
5.6.1 Hyper-parameter tuning . . . 49
5.7 Results . . . 50
5.7.1 Violence detection . . . 50
5.7.2 Multi-class classification . . . 57

6 Conclusions and future work **61**
6.1 Future Work . . . 62

A Annex A: Evaluation toolbox results **65**

B Annex B: Multi-class classification results **69**

References **71**

List of Figures

2.1 Optical flow estimation methods . . . 6
2.2 Mathematical model for the neuron . . . 8
2.3 Structural graph of a multi-layer perceptron neural network model . . . 10
2.4 Convolutional network architecture for classification . . . 12
2.5 Block diagram of the LSTM recurrent network "cell" . . . 13
2.6 Single stream network architectures . . . 19
2.7 LRCN architecture . . . 19
2.8 Two-stream network architecture . . . 20
2.9 Feature amplification with optical flow magnitude . . . 21
2.10 Three-stream network architecture . . . 21

4.1 Structure of the proposed method for audio-video group emotion recognition . . . 29
4.2 Structure of the video-based group emotion recognition module . . . 31
4.3 Structure of the audio-based group emotion recognition module . . . 32

5.1 Block diagram of the data preparation phase . . . 39
5.2 VGG16 model architecture . . . 42
5.3 Block diagram of the implementation of the LRCN architecture. . . . 43
5.4 Inflated Inception network architecture . . . 44
5.5 Occurrence errors . . . 46
5.6 Timing errors . . . 46
5.7 Segmentation errors . . . 46
5.8 Visualization tool applied to a test set video. . . . 47
5.9 Epoch progression of loss and accuracy values, optimized by the Adam algorithm. 50
5.10 Epoch progression of precision and recall values, optimized by the Adam algorithm. 50
5.11 Epoch progression of loss and accuracy values, optimized by the Adam algorithm. 52
5.12 Epoch progression of precision and recall values, optimized by the Adam algorithm. 52
5.13 False positive produced during Touching scene from a test set video. . . . 53
5.14 False negative produced during Grabbing scene from a test set video. . . . 53
5.15 Epoch progression of loss and accuracy values, optimized by the Adam algorithm. 57

A.1 Frame analysis for the violence detection algorithm, regarding the test set with the LRCN network architecture. . . . 65
A.2 Event analysis for the violence detection algorithm, regarding the test set with the LRCN network architecture. . . . 66
A.3 Frame analysis for the violence detection algorithm, regarding the test set with the I3D network architecture. . . . 67
A.4 Event analysis for the violence detection algorithm, regarding the test set with the I3D network architecture. . . . 68

List of Tables

2.1 Confusion matrix for binary classification task. 14
2.2 Confusion matrix for multi-class classification task. 14

4.1 Accuracy (%) of the proposed method on the validation set (A - only audio; V - only video; A+V - multimodal) 33
4.2 Accuracy (%) of the proposed method on the test set (A - only audio; V - only video; A+V - multimodal) 34
4.3 Confusion matrix of audio-based recognition on the validation set. 34
4.4 Confusion matrix of video-based recognition on the validation set. 34
4.5 Confusion matrix of multimodal recognition on the validation set. 35

5.1 Action statistics 38
5.2 Action distribution in train, validation and test sets 39
5.3 Aggregated classes 41
5.4 Precision, recall and F_1 score values regarding the test set for the LRCN network. 51
5.5 Confusion matrix for the test set regarding the LRCN network. 51
5.6 Precision, recall and F_1 score values regarding the test set for the violence detection task. 54
5.7 True positive rate (TPR), False negative rate (FNR), True negative rate (TNR) and False. positive rate (FPR) of the anomaly detection and action recognition algorithms. 54
5.8 Confusion matrix for the action recognition algorithm regarding the test set. . . . 54
5.9 Confusion matrix for the anomaly detection algorithm regarding the test set. . . . 55
5.10 Precision, recall and F_1 score values regarding the test set using optical flow as input. 55
5.11 Precision, recall and F_1 score values regarding the test set using a 2-stream approach. 56
5.12 Performance on test set regarding the Violence class using RGB, flow or RGB+flow as input to the model. 56
5.13 57
5.14 Classification report regarding the test set, provided by the *sklearn* library. 58
5.15 Confusion matrix for the interaction recognition task regarding the test set. 58
5.16 Classification report regarding the test set after removing the Touching class. . . . 59
5.17 Classification report regarding the test set using optical flow as input. 59
5.18 Classification report regarding the test set using a two-stream configuration. . . . 60

B.1 Confusion matrix for the interaction recognition task regarding the test set, after removing the Touching class, using RGB frames. 69
B.2 Confusion matrix for the interaction recognition task regarding the test set using optical flow as input. 69

B.3 Confusion matrix for the interaction recognition task regarding the test set, using a two-stream approach. 70

Acronyms and Abbreviations

API	Application Programming Interface
ANN	Artificial Neural Network
ACC	Accuracy
CNN	Convolutional Neural Network
ConvNet	Convolutional Network
CPU	Central Processing Unit
FP	False Positive
FPR	False Positive Rate
FPS	Frames Per Second
FN	False Negative
F_1	F_1 score
GM	Geometric Mean
GMM	Gaussian Mixture Model
GPU	Graphical Processing Unit
HSV	Hue Saturation Value
iDT	Improved dense Trajectories
IVS	Interior Vehicle Sensing
LRCN	Long-term Recurrent Convolutional Network
LSTM	Long Short-Term Memory
MFCC	Mel-Frequency Cepstral Coefficients
MLP	Multi-layer Perceptrons
MSE	Mean Squared Error
NIR	Near-Infrared
RAM	Random Access Memory
ReLU	Rectified Linear Unit
RGB	Red Green Blue
ROC	Receiver Operating Characteristic
RNN	Recurrent Neural Network
SAV	Shared Autonomous Vehicles
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
Tanh	Hyperbolic tangent
TN	True Negative
TNR	True Negative Rate
TP	True Positive
TPR	True Positive Rate
TSN	Temporal Segment Networks

Chapter 1

Introduction

Shared mobility can be defined as the shared use of a vehicle for performing a trip. Several services such as bike-sharing, scooter-sharing, car-sharing or on-demand ride services all fall into the category of shared mobility services. These services are on the rise as they can allow for a reduction in costs, provide convenience, reduce vehicle usage and vehicle ownership [1]. The growth of shared mobility services can be expedited by autonomous vehicle technology [2] while shared mobility services can enable the deployment of autonomous vehicles to be financially viable [3]. Therefore, the diffusion of shared mobility services along with emerging autonomous vehicle technology has the potential to revolutionize urban transportation.

With the arrival of Shared Autonomous Vehicles (SAV) there will no longer be a driver who is responsible for the well-being of the passengers, as well as preserving the car's interior. Without this driver's supervision, users of this service might find themselves in uncomfortable or even dangerous situations. Thus, it becomes essential for the car to have a "sense" of what is happening and of its own interior. The recognition of interactions between the passengers is crucial to understand what is happening inside the vehicle so the system should be able to understand social interactions as well as escalation levels. This can be accomplished with the use of a sensor box placed inside a vehicle to monitor all rides and with an adequate algorithm capable of detecting different types of human interactions.

Human action recognition in video is a very active research topic that has attracted a lot of attention in various application domains, such as: autonomous driving, human-machine interaction, video surveillance and health support. Despite this topic being widely studied, it is also challenging due to the high variation and complexity of the data. One of the big challenges of action recognition stems from the high variability in human actions: a single action can be performed in separate ways by different people and a person can also carry out an action in many ways.

The main goal of this dissertation is to propose and implement an algorithm that is capable of dealing with the topic of activity recognition for human interaction inside a vehicle. To achieve this goal, video footage taken with cameras inside the car will be used. Furthermore, there is also the possibility of using audio information, besides the video footage, for a more robust classification.

1.1 Contributions

The main contributions regarding this dissertation was the use of an end-to-end trainable model for human interaction recognition in the context of shared autonomous vehicles. This action recognition algorithm was trained on one of Bosch's internal datasets and benchmarked against the anomaly detection framework proposed in [4], creating a baseline for both semi-supervised and supervised algorithms on Bosch's interior human interaction proof of concept dataset. Additionally, the audio pipeline for emotion classification and respective fusion with video was developed for the EmotiW challenge, in conjunction with INESC-TEC, which resulted in a publication in an international conference.

1.2 Document structure

Besides the Introduction, this dissertation includes 5 more chapters. In chapter 2, several fundamental concepts for action recognition were studied and subsequently the state-of-art approaches were described. Chapter 3 provides us with a characterization of the problem this dissertation aims to solve, including also the proposed solution for this problem and viable alternative solutions. Chapter 4 covers the experiments that were carried out in collaboration with INESC for the EmotiW competition, specifically, for the task of audio-video group emotion recognition. The implementation details for the proposed action recognition framework are described thoroughly in chapter 5. This chapter also includes the main performance results obtained with this proposed algorithm. Lastly, chapter 6 provides the main conclusions regarding the work developed throughout this dissertation. Furthermore, some ideas on possible future work are also included in this chapter.

Chapter 2

Related Work

As previously stated in chapter 1, the main goal of this dissertation is to propose and implement an algorithm for in-vehicle human interaction recognition. However, several fundamental concepts associated with this task must first be studied and consolidated in order to understand the state-of-the-art approaches for action recognition. Therefore, this chapter is divided into these sections:

- **Tradition approaches**: small overview of the traditional approaches used to solve the problem of action recognition, 2.1.
- **Machine learning and deep learning**: definition of core concepts related to machine learning and deep learning, section 2.2.
- **State-of-the-art approaches**: description of the state-of-the-art approaches for action recognition, section 2.3.

2.1 Traditional approaches

Before 2014, the state-of-the-art techniques relied on handcrafted features to solve the problem of video-based activity recognition. There are several ways to extract visual features, including temporal and static image features, that are used to perform the classification. Temporal video information is achieved through temporal visual features, which are a combination of static image features with time information.

2.1.1 Handcrafted features

Handcrafted features represent properties derived using various algorithms that use the information present in the image itself. Edges and corners are two examples of simple features that can be extracted from an image using basic edge or corner detectors. Other complex features can also be extracted, such as pose features or motion features regarding a sequence of frames. Handcrafted features were commonly used with traditional machine learning algorithms for classification problems. With deep learning, neural networks are able to automatically extract features from the image data and learn their importance to the output. Nonetheless, handcrafted features can also be used as input in addition to or instead of the image data in order to make the classification task easier.

2.1.1.1 Optical Flow

Optical flow can be defined as the apparent motion of individual pixels on an image plane between two consecutive frames. This motion can be caused by the movement of the object or camera. Optical flow usually provides a good approximation of the true physical motion projected onto the image plane and is often a fundamental concept used in video-processing algorithms.

Most of the top performing action recognition methods use optical flow as an input. There is even the observation in literature [5] that classification accuracy, using only optical flow as input, can be higher than classification accuracy using solely raw images in some standard datasets.

Although many video categories on current datasets can be identified from a single image, this is mostly because many actions can be uniquely defined by human-object interactions (for example, playing the guitar, fencing, knitting, etc.). However, if the temporal structure is ignored and only raw images are considered, models probably will not be able to distinguish activities like opening from closing a door.

There are plenty of methods of estimating the optical flow between two frames, including differential-based, energy-based, region-based and phase-based methods. Currently, the most widely used method is based on the differential method and assumes that pixel intensities of an

object do not change between consecutive frames. More specifically: $I(x+\delta x, y+\delta y, t+\delta t)$ can be expressed with a Taylor Series expansion:

$$I(x+\delta x, y+\delta y, t+\delta t) \approx I(x,y,t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t \tag{2.1}$$

Therefore,

$$\frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t \approx 0 \tag{2.2}$$

If we divide everything by δt, we get:

$$\frac{\partial I}{\partial x}Ux + \frac{\partial I}{\partial y}Uy + \frac{\partial I}{\partial t} \approx 0 \tag{2.3}$$

The above equation is called the Optical Flow equation, where $Ux = \frac{\partial x}{\partial t}$ and $Uy = \frac{\partial y}{\partial t}$ are two components of the optical flow of the voxel (x, y, t).

As $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial x}$ are image gradients and, similarly, $\frac{\partial I}{\partial t}$ is the gradient along time, there are still two unknown variables (Ux and Uy). Since we have one equation with two unknowns for each pixel, additional constraints are required to solve for Ux and Uy. To solve this problem, several methods are provided such as the Lucas-Kanade method, the Farnebäck method and many more.

The **Lucas-Kanade method** was first proposed by Lucas and Kanade in 1981 [6]. This method computes optical flow for a sparse feature set, for example corners detected using the Shi-Tomasi algorithm. It assumes that neighboring pixels will have similar motion and, thus, takes a 3x3 patch around a point so that all 9 points have the same motion. Thus, the problem of solving one equation with two unknown variables is turned into a problem of solving 9 equations with two unknown variables, which is overdetermined. The Lucas-Kanade method obtains a compromise solution by the least squares principle. In particular, it solves the 2x2 system presented below:

$$\begin{bmatrix} Ux \\ Uy \end{bmatrix} = \begin{bmatrix} \sum_i Ix^2 & \sum_i IxIy \\ \sum_i IyIx & \sum_i Iy^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i IxIt \\ -\sum_i IyIt \end{bmatrix} \tag{2.4}$$

The **Farneback method** is considered a method for dense optical flow estimation as it was developed to output results on a dense grid of points. Dense optical flow methods have the advantage of having a higher accuracy at the cost of being more computationally expensive. Unlike the Lucas-Kanade method which only uses a sparse feature set, such as corners detected by the Shi-Tomasi algorithm, the Farneback method considers all points and detects the pixel intensity changes between the two frames. To do this, it relies on polynomial expansion to approximate each neighborhood of both frames by quadratic polynomials. Afterwards, the displacement fields are calculated from the polynomial expansion coefficients [7]. In OpenCV's implementation, the magnitude and direction of optical flow is computed from the 2-channel array of flow vectors, $\frac{\partial x}{\partial t}$ and $\frac{\partial y}{\partial t}$. Finally, the angle (direction) of flow can be visualized by hue, while the distance (magnitude) of flow can be visualized by value of HSV color representation.

(a) Lucas-Kanade method (b) Farneback method

Figure 2.1: Optical flow estimation methods

Several of the traditional approaches followed an old-fashioned pipeline, which first extracts local statistics from spatial saliencies and from action motions, then combines them into video-level representations that are, subsequently, fed into discriminative classifiers.

Currently, one of the best hand-crafted approaches is named Improved Dense Trajectories (iDT) [8]. This approach extracts features and trajectories from a dense set of interest points and encodes them in a fixed-size video description. Afterwards, a classifier (an SVM, for example) is trained on the resulting "bag-of-words" representation. However, this approach is computationally expensive and requires a lot of preprocessing to be done to each frame.

2.2 Machine learning and deep learning

Machine learning is a branch of Artificial Intelligence focused on granting systems the ability to learn and improve with experience. It is defined by Stanford University as "the science of getting computers to act in specific ways without explicitly programming them to do so." Thus, machine learning uses algorithms to parse data and learn from that data so that it can make informed decisions based on what it has learned. Machine learning algorithms are used in a wide variety of applications such as in email filtering, video surveillance, social media services and many more.

These algorithms can be broadly divided into three main categories based on the experience they are allowed to have during the learning phase [9]:

- **Supervised learning**: In supervised learning algorithms, all data is labeled, therefore, each example is associated with a label or a target. These algorithms learn to predict the output from the input data and usually require less training data than other methods.
- **Unsupervised learning**: In unsupervised learning algorithms, all data is unlabeled and the algorithm learns useful properties of the structure of the dataset.
- **Semi-supervised learning**: In semi-supervised learning, some examples include a label while others do not. A combination of supervised and unsupervised techniques can be used in this context.
- **Reinforcement learning**: Reinforcement learning algorithms do not experience a fixed dataset. Instead, they interact with an environment so that there is a feedback loop between the learning system and its experiences [9].

Deep learning is a subset of machine learning or, in other words, a special kind of machine learning, that is particularly effective in feature detection. Deep learning applications rely on a layered structure of algorithms called an **artificial neural network** (ANN). The adjective *deep* in "deep learning" refers to the layer depth of layers in a neural network. Deep learning algorithms exploit the unknown structure in the input distribution in order to determine multiple levels of representation, or a hierarchy of features, with higher-level, more abstract features defined in terms of lower-level features [10]. A key difference between machine learning algorithms and deep learning algorithms is that deep learning algorithms are capable of automatic feature engineering, while a more traditional machine learning algorithm would require the software engineer to select the relevant features beforehand. However, deep learning models also require a vast amount of training data and are usually computationally expensive to train and execute.

2.2.1 Datasets

A dataset is a collection of data or information required in data science or machine learning, where the data is usually obtained from historical observations. In the context of machine learning, datasets are used to train and evaluate a given model. There are typically three different datasets when implementing a deep neural network: the training set, validation set and test set.

The **training set** is used to fit the model so that it can learn relevant features from the training examples. The insights derived from the train set will enable the model to make better decisions. The **validation set** is used to provide an unbiased evaluation of the model fit on the training set, while tuning the model's hyperparameters. Finally, the **test set** provides an unbiased assessment of the final trained model.

A validation set is very important as it can help us identify when the network is overfitting by checking validation metrics when training the model. **Overfitting** can be detected when the

training loss is still decreasing but the validation loss starts increasing or is the same for several epochs. This happens when a model corresponds too closely to a particular set of data. An overfit model will perform extremely well on the train set, but performs poorly with new “unseen” data, meaning it has low generalization capabilities. On the other hand, **underfitting** occurs when the model is not able to achieve a sufficiently low error value on the training set. This happens when the model has not learned enough relevant patterns from the training data, which also leads to a low generalization capacity.

2.2.2 Artificial Neural Networks (ANNs)

An artificial neural network is a biologically inspired computing system that is intended to emulate the way the human brain processes information and learns. They are one of the main tools used in machine learning as they are excellent at finding patterns considered far too complex or numerous.

Neural networks are based on a collection of nodes or units connected by directed links. Each link has a numeric weight associated with it, W_{ij}, that establishes the strength and sign of the connection. A link from unit j to unit i is responsible for propagating the activation aW_j from j to i. Each neuron i will first compute a weighted sum of its inputs and then apply an **activation function** f to this sum in order to derive the output [11]. A simple mathematical model of the neuron can be observed in figure 2.2.

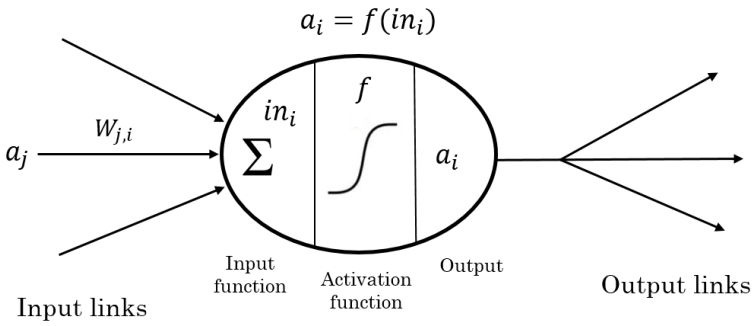


Figure 2.2: Mathematical model for the neuron, adapted from [11].

Furthermore, the neuron should be "active" or, in other words, the output should be close to 1, when the "right" inputs are given and "inactive" (output close to zero) when the "wrong" inputs are given. The most commonly used activation functions are presented below:

- **Sigmoid function**: The sigmoid activation function, also referred to as the logistic function, is a traditionally popular non-linear activation function. The inputs to this function are transformed to a value between 0 and 1. Inputs that are much larger than 1 will be mapped

to a value close to 1, while values that are much smaller than 0 will be transformed to values close to 0.

$$f(x) = \frac{1}{1+e^{-x}} \tag{2.5}$$

- **Hyperbolic tangent function (tanh)**: This activation function is similar to the sigmoid function but outputs values between -1 and 1. While sigmoid functions have been popular, the hyperbolic tangent function might be preferred, partly because it has a steady state at 0. This makes it easier to model inputs that have strongly negative, neutral as well as strongly positive values.

$$f(x) = \frac{2}{1+e^{-2x}} - 1 \tag{2.6}$$

- **Rectified linear unit function (ReLU)**: This function returns a 0 for any negative input it receives or the value provided as input directly. Hence, this function is linear for positive values, yet, it is also a non-linear function as negative values are always mapped to zero. A limitation associated with ReLU is the 'dying ReLU' problem in which a node or unit will always output the value 0. This happens when the summed input to this function is always negative, regardless of the input to the network. When this happens, a node will always output the value 0. A similar alternative to the ReLU function is the leaky ReLu activation function, which is the same as ReLU for positive values, but instead of being 0 for all negative values, it has a slight slope. This modification allows the function to output small negative values when the input is less than zero, which solves the the 'dying ReLU' problem.

$$f(x) = max(0,x) \tag{2.7}$$

- **Softmax function**: The softmax activation function can be seen as a generalization of the sigmoid function. It transforms a n-dimensional vector of arbitrary real values into a n-dimensional vector with real values in the range of (0,1) that add up to 1. Therefore, the output will be a probability distribution on classes. For this reason, this function is usually used in the output layer for neural networks whose job is to classify an input into one of multiple classes.

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \text{ for } i = 1, ..., \text{K and } \mathbf{z} = (z_1, ..., z_k) \in \mathbb{R} \tag{2.8}$$

In terms of connections inside a neural network, there are essentially two distinct ways to connect nodes together to form a network [11]. A **feed-forward network** forms a directed acyclic graph, meaning it has connections only in one direction. Thus, the information only travels forward in one direction - from the input nodes, through the hidden nodes and, finally, to the output

nodes. On the other hand, **recurrent networks** introduce loops into the network, forming a directed cyclic graph. The response of these types of networks to a certain input varies according to its initial state, which might depend on previous inputs. Thus, unlike feed-forward networks, recurrent networks are able to support short term memory.

The simplest type of feed-forward network is called the **"perceptron"**, which includes no hidden units and, therefore, all the inputs are connected directly to the outputs. A key limitation associated with single-layer feedforward networks (perceptrons) is that they can only learn linearly separable patterns as they are a type of linear classifier. Multi-layer feed-forward neural networks, also called multi-layer perceptrons (MLPs), are composed of three or more layers, including at least one hidden layer, one input layer and one output layer. A three-layer MLP is considered a shallow or non-deep neural network while an MLP with more than three layers is referred to as a deep neural network. An example of a multi-layer feed-forward network is presented in figure 2.3.

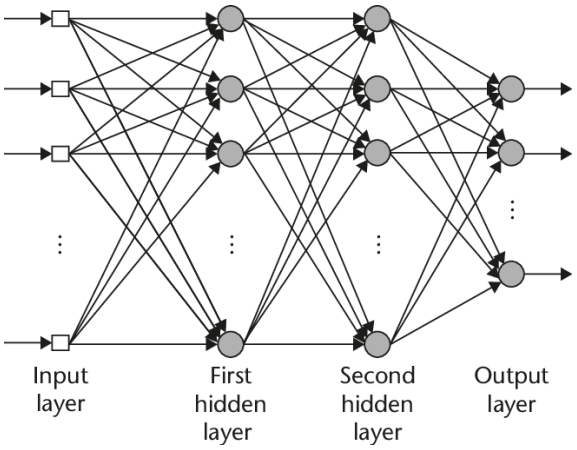


Figure 2.3: Structural graph of a multi-layer perceptron neural network model, extracted from [11].

Despite neural networks being first introduced in the 1940's, they have only become a major part of artificial intelligence in the last several decades. A major complication associated to these networks stems from the addition of hidden layers. While the error at the output layer is clear, the error at the hidden layers wasn't, as the training data does not specify which value the hidden nodes should have. It was only with a re-discovery of a technique called **backpropagation** in the 1980's [12] that neural networks blossomed. This technique allows networks to adjust the weights of their hidden layers of units by back-propagating the error from the output layer to the hidden layers. The goal of backpropagation is to minimize an error or loss function which determines how close the model's prediction is to the actual ground truth.

In terms of loss functions, there are many different types of loss functions available. Two of the most commonly used loss functions are presented below:

- **Mean squared error loss (MSE)**: The mean squared error loss is the most commonly used

loss function for regression problems. It represents the average of the squared differences between the predicted and actual values, equation 2.9

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{2.9}$$

- **Cross-entropy loss (logarithmic loss)**: Cross entropy loss is used to measure the performance of a classification model, whose output represents a probability distribution between 0 and 1. This loss increases when the predicted probability diverges from the actual target, resulting in a high loss when the difference between the prediction and ground truth is large, and vice-versa. In the case of multi-class classification, a separate loss is calculated for each class label and the results are summed.

$$CE = -\sum_{i=1}^{C} Y_i log(\hat{Y}_i) \tag{2.10}$$

2.2.3 Convolutional Neural Networks (CNNs)

Convolution Neural Networks are a vital example of a successful application of insights obtained by studying the brain to machine learning applications. They are a specialized kind of neural network for processing input data that has an inherent grid-like topology. Thus, this type of network works very well with image data and greatly outperforms neural networks on conventional image recognition tasks. They learn directly from image data, using patterns to classify images, erasing the need for manual feature extraction. Besides images, they can also be used for other domains such as audio, text, and video.

In a CNN, unlike a fully connected neural network, the neurons in one layer do not connect to all the neurons in the next layer. Instead, CNNs have sparse interactions, which are accomplished by making the kernel smaller than the input. They also use tied weights, meaning that the same feature is computed over different locations of the input. Therefore, the network will be able to correctly detect an object in an image, even if that image shifted to the right, for example. Sharing parameters across multiple locations allowed CNNs to dramatically reduce the number of unique model parameters, and to increase network sizes without requiring a corresponding increase in training data. This parameter sharing also leads to equivariant representations [9], meaning that, if the input changes (e.g., translates), the output will change similarly.

2.2.3.1 Architecture

The architecture of a convolutional neural network was inspired by the structure of the visual cortex and is analogous to the connectivity pattern of neurons in the human brain. It is a multi-layered feed-forward neural network, consisting of multiple hidden layers stacked on top of each other in sequence, besides the input and output layers. This sequential design allows CNNs to learn hierarchical features. While the initial convolutional layers learn simple and general features such as edges, the deeper convolutional layers learn more complex and high-level features.

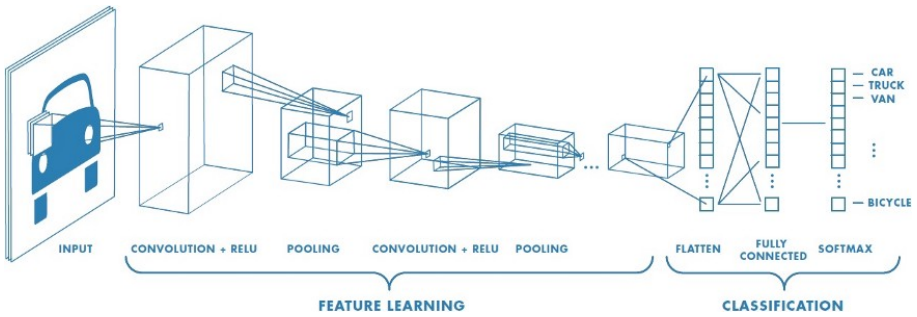


Figure 2.4: Convolutional network architecture for classification, extracted from [13].

The hidden layers are usually convolutional layers followed by activation layers and pooling layers. **Pooling layers** take the feature map projections and distill them to the most essential elements (using a signal averaging or signal maximizing process for example). Thus, pooling ensures that the network focuses on the most relevant patterns discovered. In figure 2.4, an example of a convolutional network architecture for classification is presented.

2.2.4 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks are a type of neural networks specialized for processing sequential data. They are widely used in the fields of natural language processing and speech recognition. They allow previous outputs to be used as inputs while having hidden states and, therefore, take into account historical information. The sequential information is preserved in the recurrent network's hidden state, which manages to span across several time steps as it cascades forward to affect the processing of each new example.

An RNN shares the same weights across several time steps. This parameter sharing makes it possible to extend and apply the model to examples of different forms (different lengths) and generalize across them [9]. The time step index does not necessarily refer to the passage of time in the real world, instead, it can refer only to the position in the sequence.

One key issue with RNNs is that they suffer from short-term memory, due to the **vanishing gradient problem**. Gradients are values used to update the network's weights. The vanishing gradient problem is when the gradient shrinks as it back propagates through time. Hence, if a sequence is long enough, it will be difficult for an RNN to carry information from earlier time steps to later ones.

2.2.4.1 Long-Short Term Memory Units (LSTMs)

Long-Short Term Memory Units are a variation of recurrent neural networks that were introduced as a solution to the vanishing gradient problem in [14]. They include internal mechanisms called gates that can regulate the flow of information. These gates can learn which data in a sequence is important to keep or throw away. Thus, relevant information can be passed down the long chain of sequences to make predictions. The LSTM block diagram is illustrated in figure 2.5.

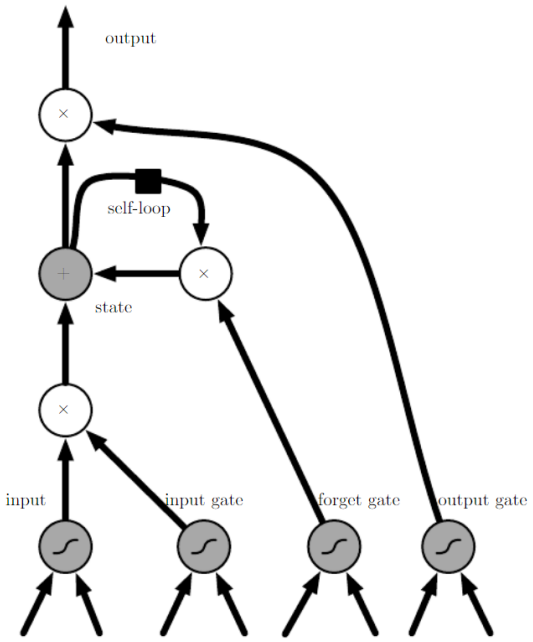


Figure 2.5: Block diagram of the LSTM recurrent network "cell", extracted from [9].

2.2.5 The classification task

Machine learning tasks are normally described in terms of how the system should process an example [9]. In this case, the example can be a collection of features regarding some event or object to be processed by the system. Some of the most common machine learning tasks include the following: classification, regression, transcription, anomaly detection and others.

The classification task requires the model to determine into which of n categories or classes a certain input sample belongs to. There are three main phases when developing a classifier: the *training phase*, the *validation phase* and the *testing phase*. During the training phase, the classification model parameters will be gradually adjusted in order to minimize the training loss. This task is usually solved in a supervised manner and can be divided into two main types: **binary classification** and **multi-class classification**.

- **Binary classification**: this type of classification task focuses on predicting one of two classes, usually referred to as the Positive class and the Negative class.
- **Multi-class classification**: this type of classification refers to predicting one of more than two classes. In the case of **multi-label** classification, each example should be assigned to one or more classes.

2.2.5.1 Evaluation

The performance of a given classifier is measured by classification performance metrics and most of them are usually scalar values like Accuracy, Sensitivity or Specificity. Additionally, there are

also graphical performance metrics such as the Precision-Recall curve and the Receiver Operator Characteristic (ROC) that are suited for measuring a classifier's performance.

The **confusion matrix** is a specific table layout that is also often used to describe the performance of a classification model. It is an *N x N* matrix, where N is the number of classes, that compares the actual target values with those predicted by the model. In the case of **binary classification**, it is a 2x2 matrix with four values, as illustrated in table 2.1.

When the classifier correctly predicts a positive sample as positive, this is known as a **True Positive** (TP). On the other hand, when a classifier incorrectly predicts a positive sample as a negative one, this can be referred to as a **False Negative** (FN). Aditionally, a **False Positive** (FP) happens when a negative sample is incorrectly classified as positive and a **True Negative** refers to the case where the negative sample is correctly classified as negative.

In the case of multi-class classification, the confusion matrix is slightly different as there are more than 2 classes for the model to predict. Unlike in the binary confusion matrix, there are no positive and negative classes and the True Positives value is associated to each individual class. An example of a confusion matrix for a multi-class classification problem with three classes is presented in table 2.2. Thus, the number of True Positives for class 1 (TP_1) is the number of samples from class 1 that were correctly classified as class 1. Additionally, the false negatives for class 1 are the result of the sum of F_{12} and F_{12}, which represent the number of samples from class 1 that were incorrectly predicted as class 2 or class 3 respectively. In fact, the number of false negatives for each class can be calculated by adding all the values of the row corresponding to that class except for the True Positives value. The same reasoning can be applied to the number of false positives for each class, where the values of the column corresponding to that class should be added except for the True Positives value.

Table 2.1: Confusion matrix for binary classification task.

		Predicted values	
		Positive (P)	Negative (N)
Actual values	Positive (P)	True Positives (TP)	False Negatives (FN)
	Negative (N)	False Positives (FP)	True Negatives (TN)

Table 2.2: Confusion matrix for multi-class classification task.

		Predicted values		
		Class1	Class 2	Class 3
Actual values	Class 1	TP_1	F_{12}	F_{13}
	Class 2	F_{21}	TP_2	F_{23}
	Class 3	F_{31}	F_{32}	TP_3

Most of the standard scalar metrics used for classification problems can be computed directly or indirectly from the confusion matrix, including the following:

- **Accuracy**: Accuracy is often used to measure the performance of a given classifier. This metric refers to the number of samples that were correctly classified out of the total number of samples:

$$ACC = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{2.11}$$

In the case of binary classification, it can be defined in terms of positives and negatives in the following way:

$$ACC = \frac{TP+TN}{TP+TN+FP+FN} \tag{2.12}$$

Accuracy can be misleading when dealing with imbalanced datasets. For example, when there are very few samples belonging to the positive class and a high number of negative samples, a given classifier might yield a very high accuracy by always predicting the negative class. Another problem with accuracy is that two different classifiers can achieve the same accuracy, yet perform very differently in terms of the type of correct or incorrect predictions they provide.

- **Recall or Sensitivity**: Recall, sensitivity or True Positive Rate (TPR) is related to the number of samples of a given class that were correctly identified. It measures the proportion of positive samples that were correctly predicted, equation 2.13

$$Recall = \frac{TP}{TP+FN} = \frac{TP}{P} \tag{2.13}$$

- **Precision**: Precision can be defined as the fraction of samples of a given class that were correctly predicted out of all the instances where the model predicted that class, equation 2.14.

$$Precision = \frac{TP}{TP+FP} \tag{2.14}$$

- **F-score**: The F-measure provides us with a weighted average of the precision and recall values. The standard F-score, also known as F_1 score or balanced F-score, provides us with a harmonic mean of the precision and recall. A more general F-score, F_β, uses a positive real factor β that allows us to weight precision or recall more highly depending on how important they are for our use case, equation 2.15. For examples if, β is = 2 (F_2 score), recall is considered twice as important as precision. Hence, in a scenario where a false negative has a different cost than a false positive, the F_1 score is not the most appropriate measure and the F_β score should be used instead.

$$F_1 = 2 * \frac{precision * recall}{precision + recall} \tag{2.15}$$

$$F_\beta = (1+\beta)^2 * \frac{precision * recall}{(\beta^2 * precision) + recall} \tag{2.16}$$

- **False positive rate**: The false positive rate, also known as false alarm rate or probability of false detection, represents the proportion of negative cases that were incorrectly identified as positive cases in the data. It is defined in equation 2.17 as the number of false positives divided by the total number of negative examples.

$$FPR = \frac{FP}{FP+TN} = \frac{FP}{N} \tag{2.17}$$

- **True negative rate or Specificity**: The true negative rate, Specificity or inverse recall, measures the proportion of negative examples that were correctly classified as negative. It is the complement of the false positive rate and can be computed with the following equation 2.18:

$$TNR = 1 - FPR = \frac{TN}{FP+TN} = \frac{TN}{N} \tag{2.18}$$

- **False negative rate**: The false negative rate or miss rate, focuses on the negative examples and is related to the number of positive samples that were incorrectly classified as negative, equation 2.19.

$$FNR = 1 - TPR = \frac{TN}{FP+TN} = \frac{TN}{N} \tag{2.19}$$

- **Geometric mean**: The geometric mean, as the name suggests, provides us with a geometric mean of the sensitivity (TPR) and specificity (TNR) values, as illustrated in equation 2.20.

$$GM = \sqrt{TPR * TNR} = \sqrt{\frac{TP}{TP+FN} * \frac{TN}{FP+TN}} \tag{2.20}$$

2.2.5.2 Class imbalance

An **imbalanced classification problem** can be defined as a type of classification problem where the distribution of classes is biased or skewed, which happens when the number of examples in the training set for each class label is not balanced. "Any dataset with an unequal class distribution is technically imbalanced. However, a dataset is said to be imbalanced when there is a significant, or in some cases extreme, disproportion among the number of examples of each class of the problem." [15]. A slight imbalance among the classes is usually not a concern and the problem can be treated as a normal classification problem. Nevertheless, a severe imbalance can pose a challenge for predictive modelling and might require the use of specialized techniques.

With a severe imbalance among the classes, the minority classes are harder to predict as there are much fewer examples of these classes. Hence, it is more challenging for the model to learn relevant features from these examples and to distinguish them from examples from the majority classes. In fact, as most machine learning algorithms were designed to maximize accuracy and reduce errors, a model might achieve a very high accuracy by only predicting the majority classes and overlooking the minority classes. Thus, training a model with severely imbalanced data usually results in a model with a poor predictive ability for the less frequent classes [15].

2.3 State-of-the-art approaches

In this section, an overview is given of the state-of-the-art approaches to solve the problem of action recognition in video. It builds upon some core concepts described in the previous section and applies them in the context of video input data.

2.3.1 Standard action recognition datasets

As previously mentioned in section 2.2.1, datasets are used to train and evaluate a given model. The size of the dataset is very important as complex problems require large and complex models, which are more likely to overfit with limited training data. Therefore, complex problems require deep networks which, in turn, need a vast amount of data to learn complex relevant patterns.

Classification problems, such as the action recognition problem, are usually solved in a supervised manner, meaning that each example in the dataset is associated with a label. This section describes several of the landmark datasets used for action classification purposes.

2.3.1.1 UCF-101

The UCF-101 is an action recognition dataset of realistic action videos collected from YouTube [16]. It includes 13 320 videos from 101 different action categories. The videos in 101 action categories are grouped into 25 groups, where videos from the same group may share some common features like similar background, similar viewpoint, etc.

2.3.1.2 HMDB51

The HMDB51 is a video database for human motion recognition and contains 51 different action categories [17]. Each category contains at least 101 clips from a total of 6,766 video clips that were extracted from a broad range of sources. This dataset includes 7 human interaction classes such as *hug*, *kiss*, *punch* or *shake hands*.

2.3.1.3 Kinetics-700

The Kinetics-700 is an extension of the Kinetics dataset, introduced in [18], and is composed of 700 video classes with approximately 600 videos per class. It is formed by a collection of clips from YouTube videos and includes 11 human-human interaction classes such as the *handshake*, *hug* and *massage feet* classes. Each clip lasts around 10 seconds and is annotated with a single action class label. This dataset features a large variety of background clutter, illumination settings and motion blur.

2.3.1.4 Moments in Time

The Moments in Time dataset is a large-scale dataset for recognizing and understanding action in videos [19]. It includes a collection of one million labeled 3 second videos, involving people, animals, objects, or natural phenomena. This dataset was designed to have large inter-class and intra-class variation that represent dynamical events at different levels of abstraction.

2.3.2 Deep-learning approaches

Deep learning architectures prevailed with state-of-the-art performance on standard video action recognition datasets such as UCF101, Sports-1M and HMDB51. In 2014, two important papers were published that prompted the popularity of single stream and two stream networks in action recognition: Large-scale Video Classification with Convolutional Neural Networks by Karpathy et. al. [20] and Two-Stream Convolutional Networks for Action Recognition in Videos by Simonyan and Zisserman [21].

While handcrafted features are still widely used for human action representation, deep learning is being actively explored for human action recognition. Based on the success of applying deep learning techniques to image-level classification tasks, recent works have been using similar learning-based representations for action recognition in video. Some of the key differences between video classification architectures include whether the input to the network is just the RGB video or if it includes additional handcrafted features (e.g. two-stream or three-stream approach), whether the convolutional and layers operators use 2D or 3D kernels, and how information is propagated across frames (in the case of 2D ConvNets).

2.3.2.1 Convolutional networks

The high performance of CNNs on image classification tasks make it appealing to try to reuse them for video. This can be accomplished by extracting features independently for each frame and pooling their predictions across the whole video [20]. However, there is the disadvantage of the model completely ignoring temporal information and, therefore, it will not be able to differentiate actions like going up from going down the stairs.

In [20], several approaches were also explored for fusing information along the temporal dimension through the network, including early fusion, late fusion, and slow fusion. In the early fusion method, frames are stacked as channels and video descriptors are learned through 2D convolution on the entire stack of frames. Late fusion places two separate single-frame networks several frames apart and the two streams are merged at the densely connected layers. Finally, the slow fusion method seeks to concatenate features in a hierarchical fashion from a stack of frames, so that higher layers will get access to progressively more global information in both space and temporal dimensions. These methods are illustrated in figure 2.6, extracted from [20].

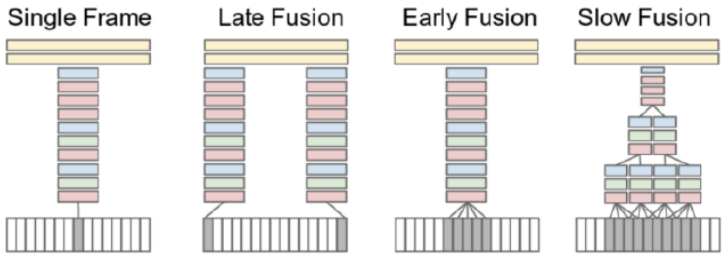


Figure 2.6: Single stream network architectures explored in [20]. Red, green and blue boxes indicate convolutional, normalisation, and pooling layers respectively. In the Slow Fusion model, the depicted columns share parameters.

While 2D ConvNets have shown remarkable results on image related tasks, they lack the ability to recognize patterns across time. Therefore, a recurrent layer, such as an LSTM, which can encode state and capture temporal ordering as well as long range dependencies, can be added to the model. Recurrent Neural Networks have been effectively used as a supplementary architecture to CNNs for extracting temporal features several times in the past [22], [23], [24]. In these architectures, the CNNs are responsible for extracting spatial information that is then passed to recurrent networks for learning the temporal characteristics associated with each class.

In [22], the authors proposed a Long-term Recurrent Convolutional Network (LRCN) model combining a deep hierarchical visual feature extractor (a CNN) with an LSTM. For activity recognition purposes, the LRCN is trained to predict the video's activity class at each time step, as shown in figure 2.7. To produce a single label prediction for the whole video, the label probabilities are averaged, and the most probable label is chosen.

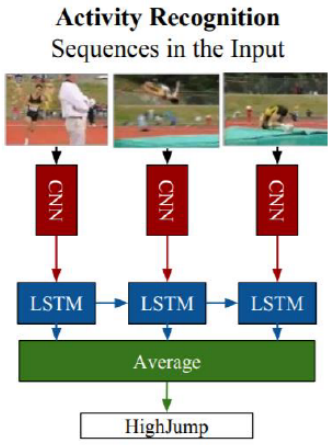


Figure 2.7: LRCN architecture proposed in [22].

2.3.2.2 Multi-modal networks

K. Simonyan and A. Zisserman [21] introduced a two-stream network architecture for action recognition in video. This architecture was designed to mirror the pathways of the human visual cortex for object recognition and motion detection. It is composed of two streams, as shown in figure 2.8. One stream uses a network focused on learning spatial features from RGB images, while the other stream uses a network focused on learning temporal features from optical flow inputs.

The spatial stream ConvNet operates on the individual image frames, performing action recognition on still frames. The input to the temporal stream ConvNet model is formed by stacking optical flow displacement fields between several consecutive frames. As such input already describes the motion between video frames, the recognition process is easier as the network does not need to estimate motion implicitly. The final classification is computed by combining softmax scores (the outputs) of the spatial network and temporal network by late fusion.

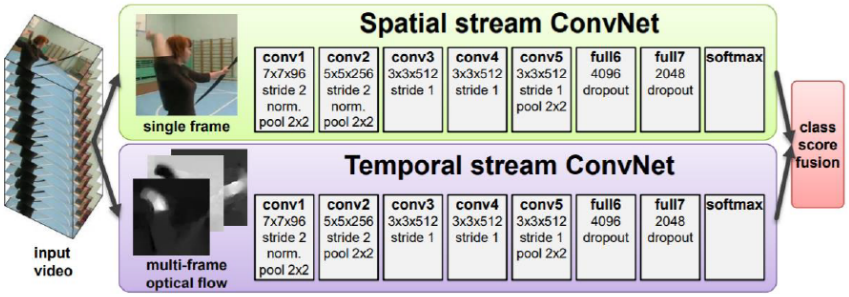


Figure 2.8: Two-stream network architecture proposed in [21].

An extension of this approach [25] explored different ways of fusing the two networks over space and time, in order to best exploit this spatial-temporal information. In this extension, the two networks are fused in the last convolutional layer into the spatial stream to convert it into a spatial-temporal stream. Furthermore, this architecture does not significantly increase the number of parameters, yet exceeds the previous state-of-the-art techniques on standard datasets.

In [26] the authors proposed a new way for combining different sources of knowledge in deep learning. Inspired by the two-stream network architecture introduced in [21], the authors proposed a way of enhancing each network with knowledge from the other stream. This approach includes feature amplification, where an auxiliary, hand-crafted feature (e.g. optical flow) is used to perform spatially varying soft-gating on intermediate CNN feature maps. Their proposed feature amplification is illustrated in figure 2.9. A multiplicative fusion method is also presented for combining CNNs trained on different sources, resulting in a more robust prediction.

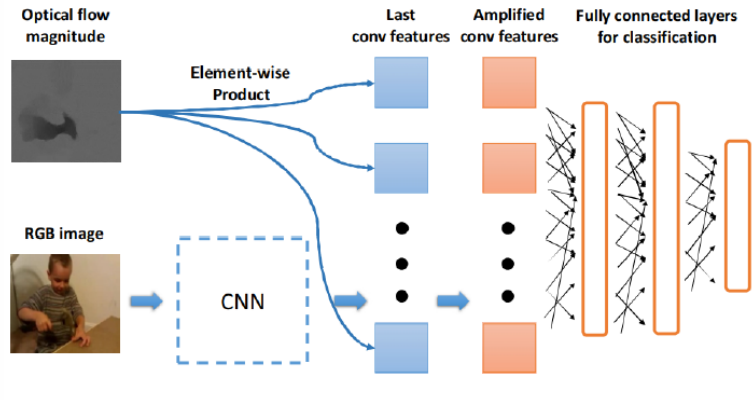


Figure 2.9: Feature amplification with optical flow magnitude proposed in [26].

Inspired by the successful two-stream networks for action classification, Muhammad Khalid, and Jie Y in [27] proposed a multi-modal three-stream network for action recognition, where additional pose features were studied and fused. Specifically, they proposed an end-to-end CNN framework to train directly on body joint tensor, which can be derived from Ground Truth poses or pose detections by recent pose estimators. For the Pose Stream, they used a ConvNet that is trained in an end-to-end fashion with Pose Tensor. The Temporal Segment Network (TSN) framework proposed in [28] was used for training the RGB and optical flow streams. The three-stream convolutional neural network (TSCNN) is drafted by fusing the scores from the pose stream with the scores from spatial and temporal streams of TSN with a weighted sum. In figure 2.10, extracted from [27], an overview of their proposed three-stream network architecture is presented.

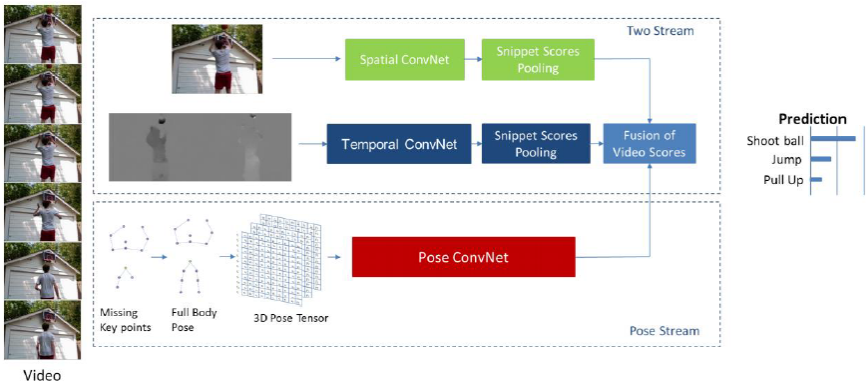


Figure 2.10: Three-stream network architecture proposed in [27].

2.3.2.3 3D ConvNets

3D convolutional networks are very similar to standard convolutional networks but include spatio-temporal filters. These types of networks have been explored several times in the past [20],[29],[30] as they create hierarchical representations of spatio-temporal data. One issue with 3D ConvNets is the high number of additional parameters compared to 2D ConvNets, due to the additional kernel dimension. Therefore, these networks are harder to train and require a high amount of training data.

Another way of learning spatiotemporal relationships is proposed in [31], where the authors introduced Factorized Spatio-Temporal Convolutional Networks (FSTCN). These networks factorize the original 3D convolution kernel learning as a sequential process of learning 2D spatial kernels in the lower layers (spatial convolutional layers), followed by learning 1D temporal kernels in the upper layers (temporal convolutional layers). This factorization design mitigates the difficulty of training networks with high-kernel complexity and with limited training data. There are several recent approaches that also decompose the convolutions into separate 2D spatial and 1D temporal filters, such as in [32], [33], [34].

In [35], the Two-stream Inflated 3D ConvNet (I3D) was introduced. This network is based on 2D ConvNet inflation, where filters and kernels of very deep image classification ConvNets are expanded to 3D. The C2D filters can be adapted to C3D by replicating them along the temporal axis. Thus, their proposed model leverages the successful ImageNet architecture designs as well as their parameters. While 3D ConvNets can directly learn temporal patterns from an RGB stream, the authors demonstrated that their performance can be improved by including an optical-flow stream. Their I3D model was based on the ImageNet pre-trained Inception-V1 [36] and they also experimented with pre-training on the Kinetics dataset mentioned in section 2.3.1. Their model was able to significantly outperform its predecessors in action classification on landmark datasets, after pre-training on Kinetics.

2.3.2.4 Context-related approaches

The deep learning-based approaches mentioned above are used to solve the problem of action recognition in video. However, there is almost no research available regarding action recognition in the context of Shared Autonomous Vehicles. Therefore, this section explores some deep learning approaches for solving the action recognition problem in closely related contexts.

In [37], a driver activity recognition system for intelligent vehicles based on deep CNNs was proposed. Several different pre-trained CNN models were adopted and evaluated, namely, the AlexNet [38], GoogleLeNet [36] and ResNet50 [39]. In this approach, the experimental images were collected using a low cost RGB camera located inside the car. Image segmentation is performed with the Gaussian Mixture Model (GMM), which is used to extract the driver body from the background before training the model. They found that the classification accuracy on driving

activities could be significantly improved by applying a segmentation model to process the raw images before the behavior detection network.

In [40], an in-vehicle driver and passenger activity recognition system was proposed. This system seeks to classify individual activities as well as human-object interactions and human-human interactions. Multiple RGB-IR cameras with combined viewing covering all vehicle seat positions will be used to accomplish this.

In this proposed architecture, the input sequences of images are down-sampled, converted to grayscale and fed into three parallel processing modules. The body posture recognition module provides skeleton model estimations of all people present inside the car. The object recognition module provides object classification and related 2D locations of relevant objects. The optical flow module produces optical flow images that capture the local movements of image pixels. They also considered image stabilization and image background subtraction so that only the relevant motions related to the activities of interest are captured.

The extracted features from these modules are then concatenated into a fixed-length feature vector for each time step, which is, subsequently, fed into a set of densely connected layers. The last module in the chain is an LSTM layer in order to detect more complex activities. This proposed architecture is still a prototype as there are no publicly available datasets for training of driver and passenger in-vehicle activities. Thus, the authors only performed a preliminary validation of the system performance using data captured in a simulation environment with limited objects and activities.

2.3.2.5 Automotive Interior Sensing

In [4], an anomaly detection framework was developed for the context of shared autonomous vehicles. This framework is based on a semi-supervised spatiotemporal autoencoder framework, proposed in [41], which hinges on the reconstruction error. The autoencoder is used to learn regularity in video sequences in a way that the reconstruction of regular video sequences results in a low reconstruction error, and the reconstruction of irregular (abnormal) video sequences results in a high reconstruction error. Furthermore, this framework outputs a regularity score in which values closer to 0 represents abnormal frames, while values closer to 1 represent normal frames. This framework has the advantage of using a semi-supervised learning approach, however, it can only detect abnormal actions and is not capable of distinguishing them from each other.

Chapter 3

Problem Description

3.1 Problem definition

Although there has been significant progress in the research of human activity recognition, the recognition of human interactions from video remains a challenging topic. This is partly because understanding human interactions requires more than the analysis of each person's action in isolation. Instead, the true nature of the interaction is revealed by the coordination, in both space and time, between people [42]. Furthermore, the context in terms of who is interacting, why and where this interaction takes place affects how it unfolds.

The problem this thesis aims to solve can be defined as a multi-class classification problem of human-human interactions in the context of Shared Autonomous Vehicles. As previously mentioned in section 2.2.5, multi-class classification refers to classification tasks that include more than two class labels. Therefore, examples are classified as belonging to one class among a range of known classes, based on the experience gathered while training the model. There are many different human interactions, however, this dissertation will focus only on possible interactions between passengers inside a vehicle. Human interactions in crowded scenes are also not considered as they are outside the scope of this thesis.

In the context of interaction recognition inside a vehicle, there will be several aspects that should be considered when implementing a deep neural network for this topic. For example, the lighting inside the car will be very different during the day and at night and the model should still work in a setting with limited illumination. There are also different shadows that might appear in the video footage while the car is moving that might confuse the algorithm. Finally, the passengers will be close together in a confined space so several of their body parts might be occluded one time or another. In fact, back-seat interactions are especially affected due to front-seat occlusions or interactions that might be partially blocking the field of view of a single-camera system.

3.2 Proposed solution

The proposed solution for this problem relies on a modular framework, developed as part of the work on this thesis, for the data preparation phase, which allows for easily switching between datasets or models. Furthermore, the proposed solution is based on two types of network architectures that are commonly used to solve the problem of action recognition in video: a convolutional network with recurrent network elements and a 3D convolutional network.

As 2D convolutional networks have shown remarkable results in image-related tasks, they can be used as spatial feature extractors for RBG images along with a recurrent network layer that is responsible for extracting temporal features. On the other hand, 3D convolutional networks include spatio-temporal filters and, therefore, are able to learn temporal patterns directly from an RGB stream. As previously mentioned in section 2.3.2.3, a problem with 3D convolutional networks is the high number of additional parameters, which make these networks harder to train. However, in [35], where the Inflated Inception network (I3D) was introduced, a major contribution was the possibility of the use of transfer learning from the Kinetics dataset, described in 2.3.1, to other datasets for simliar tasks. Transfer learning allows for a reduced training time by re-using model weights from pre-trained models that were developed for related tasks. Therefore, the proposed solution includes using transfer learning with the I3D model proposed in [35] pre-trained on ImageNet and Kinetics datasets.

The performance of both networks will then be compared with the help of an evaluation toolbox provided by Bosch for the task of violence detection. The network architecture exhibiting a superior performance on this task will be selected for the more complex task of interaction recognition.

3.3 Alternative solutions

An alternative solution for this problem can be based on the successful two-stream networks used for action recognition purposes. These architectures usually rely on additional handcrafted features such as pre-computed optical flow. However, optical flow might not be the best option in this context, considering the unwanted camera vibrations, caused by the motion of the vehicle, that can be picked up by the algorithm. The motion of the shadows, caused by different illumination settings when the car is moving, can also confuse the optical flow algorithm.

Incorporating additional pose features has also demonstrated an improved performance in action recognition systems. Furthermore, recent pose estimators usually achieve good results even with poor lighting conditions. Thus, pose features should be further explored and fused to check if a similar boost in performance will happen in this context.

It would be beneficial for the model to focus only on the people present in the scene as this is an interaction recognition problem. Therefore, a background subtraction technique, like Gaussian Mixture Model (GMM), could be used in order to accomplish this. Recent results also indicate that appropriate image segmentation positively impacts performance in similar problems [37].

Chapter 4

Emotion Recognition

This chapter offers a thorough description of the experiments performed regarding classification in the video and audio domain for a public dataset. These were carried out in collaboration with INESC TEC for the EmotiW2020 challenge, particularly for the task of "Audio-video Group Emotion Recognition". Furthermore, this chapter also explores the advantage of using audio information, through score-level fusion, for an improved performance regarding classification tasks. This chapter is divided in the following sections:

- **Introduction**: context and motivation for the task of group emotion recognition, 4.1.
- **Related Work**: literature review of the state-of-the-art approaches for group emotion recognition, section 4.2.
- **Implementation**: details regarding the main steps the implementation of the proposed solution, section 4.3.
- **Results**: details regarding the network architectures used for the task of violence detection as well as class imbalance countermeasures, 4.4.
- **Conclusion**: details regarding the network architectures used for the task of violence detection as well as class imbalance countermeasures, 4.5.

4.1 Introduction

Emotion recognition is a fast-growing research topic, due to its potential for enhanced human-computer interfaces and automatic services that immediately respond to the emotions of the user or client [43].

The task of group emotion recognition shares some similarities with the recognition of human activity based on the video. Unlike the former, the latter boasts several large and thoroughly labeled datasets, such as the Kinetics or the ActivityNet [44], even when restricting to data focused on groups rather than on individuals.

Hence, this chapter explores the novel application of inflated convolutional neural networks (CNN) to classify emotion valence at the group level in videos. The use of audio for improved performance was also studied, through score-level fusion, with a Bi-LSTM network receiving spectral features. Throughout the experiments, the performance of the proposed method for multimodal and unimodal classification was assessed and compared directly with the EmotiW 2020 subchallenge official baseline.

4.2 Related Work

State-of-the-art methods for emotion recognition are mainly based on facial expressions, and important hurdles have been overcome in this field [43], [45]. Group-level emotion is a fairly uncharted research topic that extends the analysis to the emotional state displayed by a group of people as a whole [46]. This topic was the focus of the EmotiW 2020 [47] subchallenge that motivated this work. The scarce data and the difficulty in obtaining annotations is the reason why few have addressed this topic [48], [49], [50], and why current approaches still offer low accuracy levels.

While methods based on visual information compose most of the literature for video classification tasks, some works discuss the advantages of including additional sources of information, especially audio [51] , [52], [53], [54]. Specifically, it has been shown that using audio complements some of the flaws of video-based recognition [51], despite offering subpar accuracy results when in a unimodal recognition system. These results have confirmed the advantages of combining audio information with a strong method for video-based recognition.

4.3 Implementation

The proposed algorithm, illustrated in figure 4.1, is composed of three modules[1]: a video-based emotion recognition model, an audio-based emotion recognition model, and a multimodal fusion module. Video and audio-based emotion recognition modules are trained separately, while the fusion module, based on a multiclass Support Vector Machine (SVM), receives the softmax scores provided by the other two.

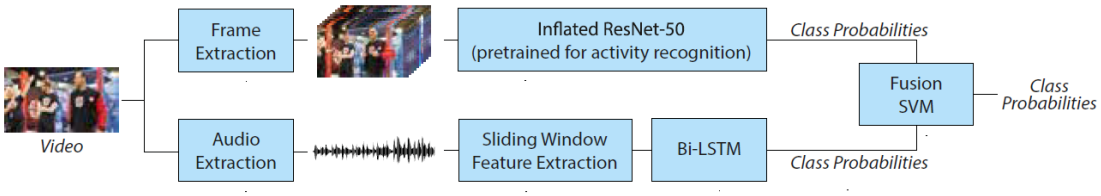


Figure 4.1: Structure of the proposed method for audio-video group emotion recognition, extracted from [55].

4.3.1 Dataset

All the experiments were conducted on an adapted version of the "Video-level Group AFfect" (VGAF) dataset [56] for the EmotiW 2020 AV Group-level sub-challenge [47]. The VGAF is a video-based database that contains labels for emotion and cohesion. The data was collected from the YouTube platform and consists of videos under the creative commons license (CC0) and present keywords that correspond to the range of emotions and cohesion.

For the EmotiW 2020 AV Group-Level Emotion subchallenge, the task was the classification of group emotion. The VGAF dataset videos were divided into five-second videos: 2661 for the train, 766 for the validation, and 756 for the test. Each video (except those in the test set) is accompanied by a discrete group emotion valence ground-truth label. In the training dataset, there are 802 positive videos, 923 neutral videos, and 936 negative videos. On the other hand, the validation set included 302 positive videos, 280 neutral videos, and 184 negative videos.

[1]The work done as part of this solution was mainly focused on the audio pipeline research, development and implementation as well as contributing in the model's fusion.

4.3.2 Baseline

The baseline algorithm is the audiovisual group-level emotion recognition sub-challenge baseline [56] of the EmotiW 2020 Grand Challenge. This method is composed of two streams, for audio and video data processing, fused at the feature-level. The video stream is a pretrained Inception V3 network that separately processes frames extracted from a video. The extracted features are combined using a long short term memory (LSTM) network. The audio stream is composed of a fully-connected network that receives OpenSMILE [57] features extracted from the audio. The outputs from the video and audio streams are concatenated and used by a fully-connected layer to offer two outputs: the probabilities for the three emotion valence classes and an emotion cohesion value on the [0-3] range.

4.3.3 Video-based emotion recognition module

The video-based emotion recognition module is based on an inflated bidimensional (2D) convolutional neural network (CNN), similar to I3D [35], the state-of-the-art in activity recognition. The model is an end-to-end network: it receives frames extracted from a video, ordered and concatenated over a time dimension, and returns class probabilities for that video. The architecture of the network follows the structure of a ResNet-50, proposed by He et al. [39].

This model has been pre-trained to discriminate between 339 activity classes on the Multi-Moments In Time database, described in section 2.3.1. To offer probability outputs for each of the three group-level emotion valence classes, the last fully-connected layer of this network is replaced by a three-neuron fully-connected layer, followed by softmax activation, as shown in figure 4.2.

4.3.4 Audio-based emotion recognition module

The audio-based recognition module, presented in figure 4.1, is composed of two main processes: feature extraction on sliding windows, and a Bi-LSTM recognition model. Audio features were extracted using the *pyAudioAnalysis* library which contains an off-the-shelf feature set. All these features were extracted over sliding windows of 25 milliseconds with a time step of 10 milliseconds.

The features are received by a Bi-LSTM model with local attention, adapted from [58], that returns the class probabilities for the respective audio, as shown in figure 4.3. Its weighted-pooling strategy enables the focus on the specific sound parts which contain strong emotional characteristics, controlled by an attention function trained simultaneously with the Bi-LSTM model.

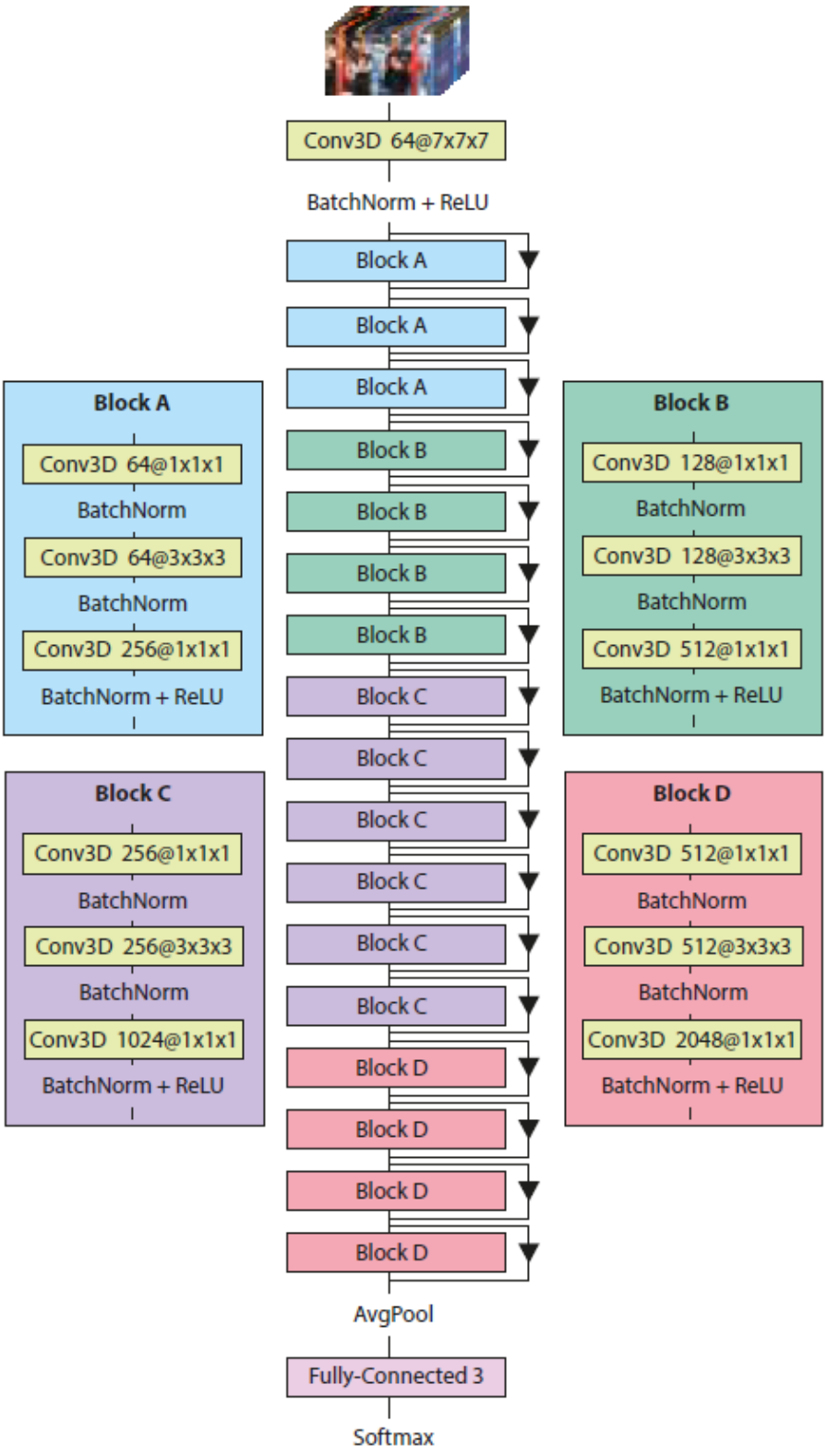


Figure 4.2: Structure of the video-based group emotion recognition module, based on an inflated ResNet-50, extracted from [55].

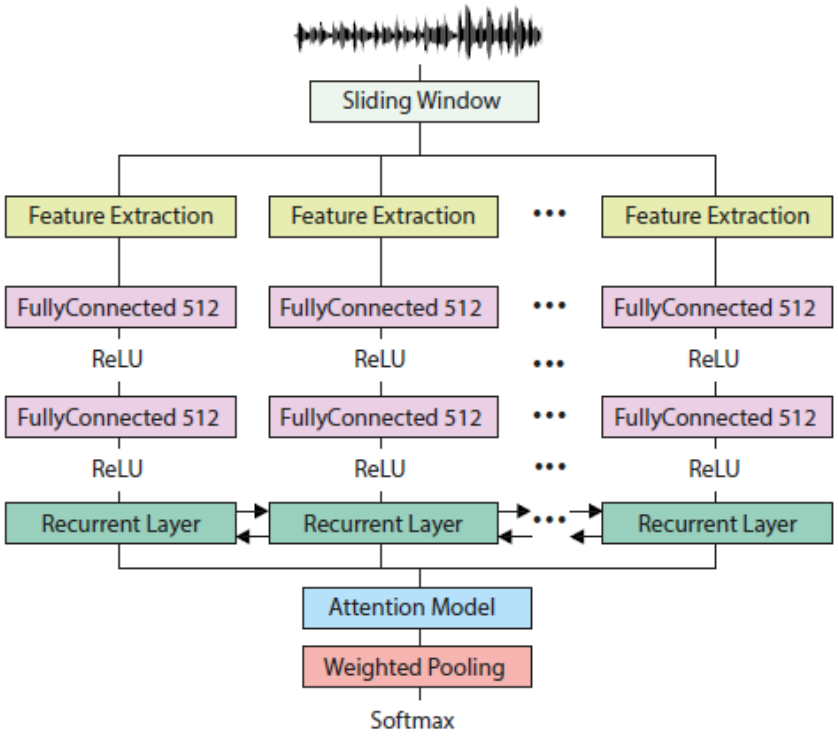


Figure 4.3: Structure of the audio-based group emotion recognition module, based on an Bi-LSTM network model, extracted from [55].

4.3.5 Score-level ensemble

The softmax scores obtained by both video and audio based emotion recognition models are, subsequently, concatenated in a feature vector. Finally, this feature vector, composed of six class probability values, is given to a multi-class SVM that combines the separate audio and video predictions into a single decision.

4.3.6 Training

The audio-based recognition module was trained from scratch. The weights were randomly initialized, and the model was trained over a maximum of 200 epochs, with a batch size of 128, categorical-cross-entropy as the loss function, and using the Adam optimizer with an initial learning rate of 1e-2. To prevent overfitting, a dropout layer was included and early-stopping was used with a patience value of 10 epochs.

The video-based recognition module, pretrained on the MMIT database, was adapted to output probabilities for each of the three valence classes in group emotion recognition.

When training the audio-based module and the video-based module, the hyperparameters were selected empirically in order to maximize performance in the validation set. On the other hand, the hyperparameters for the fusion module (e.g., the regularization parameter “C”, the kernel,

or the polynomial degree of the kernel) were found through a grid-search. Once the optimal hyperparameters were found, the train and validation set were combined to make a "full train" set, and therefore take full advantage of all available labeled data for better performance in the test set.

4.3.7 Evaluation

The accuracy metric was used along with confusion matrices to examine the overall performance of the method and also analyze its class-wise accuracy. The performances of the multimodal method and its audio and video-based modules were, separately, evaluated in both the validation set (with available groundtruth labels) and the test set (accuracy values delivered by the EmotiW 2020 sub-challenge organization upon request). Finally, the performance was compared with the official sub-challenge baseline, following the results reported in the paper [56].

4.4 Results

The performance results of the proposed method on the validation and test sets are presented, respectively, in tables 4.1 and 4.2. These tables also include the official sub-challenge baseline results for comparison.

Table 4.1: Accuracy (%) of the proposed method on the validation set (A - only audio; V - only video; A+V - multimodal)

	Accuracy			
Method	Overall	Positive	Neutral	Negative
Proposed (A)	47.12	19.93	**69.64**	57.61
Proposed (V)	**62.40**	69.20	52.50	66.30
Proposed (A+V)	**61.83**	58.13	61.78	**67.93**
Baseline (A)	50.23	-	-	-
Baseline (V)	52.09	-	-	-
Baseline (A+V)	**50.23**	-	-	-

On the validation set, the performance offered by the proposed multimodal method is superior to the baseline. The accuracy attained by the video-only approach is close to that offered by the multimodal method, over 62%. This is evidence of the advantages of using pretrained networks (in this case, transferred from the task of human activity recognition). The audio-only approach offers considerably lower performance (47%) than the audio-only baseline (50%), which indicates the use of OpenSMILE [57] features and fully-connected networks may be better fitted for group emotion recognition based on audio.

From the validation to the test set (table 4.2), the official video-only baseline suffers a sharp performance decay (from 52% to 42% accuracy), which is also felt with the proposed video-only approach (albeit not as dramatic, from 62% to 59% accuracy). Fusing with audio on a multimodal

Table 4.2: Accuracy (%) of the proposed method on the test set (A - only audio; V - only video; A+V - multimodal)

	Accuracy			
Method	Overall	Positive	Neutral	Negative
Proposed (V)	58.86	55.76	57.93	63.04
Proposed (A+V)	**65.74**	54.38	**77.99**	60.00
Baseline (V)	42.00	-	-	-
Baseline (A+V)	47.88	45.00	10.00	70.00

approach reduced that decrease in the case of the official baseline (from 50% to 48% accuracy), and even reversed it in the case of the proposed method (from 62% to 66% accuracy). This confirms the idea present in the literature that, while audio alone is not suitable for recognition, it offers additional information that is essential for the robustness and accuracy of the method.

Analysing the class-wise accuracies and the confusion matrices (tables 4.3, 4.4, and 4.5), one can notice that video is, overall, the best modality to recognise emotions. The advantages of using video rely mainly on the "extreme" classes, positive and negative, which denote visual information is more advantageous to recognize strong group emotions. In contrast, the proposed audio-only approach attains very poor accuracy in the positive class.

Table 4.3: Confusion matrix of audio-based recognition on the validation set.

		Predicted values		
		Positive	Neutral	Negative
Actual values	Positive	60	139	102
	Neutral	35	195	50
	Negative	24	54	106

Table 4.4: Confusion matrix of video-based recognition on the validation set.

		Predicted values		
		Positive	Neutral	Negative
Actual values	Positive	209	59	34
	Neutral	98	147	35
	Negative	44	18	122

Table 4.5: Confusion matrix of multimodal recognition on the validation set.

Actual values \ Predicted values	Positive	Neutral	Negative
Positive	175	99	27
Neutral	78	173	29
Negative	27	32	125

4.5 Conclusion

In this chapter, a novel method for the automatic recognition of group emotion was proposed. This method uses a late-fusion multimodal approach, combining scores from both video and audio based emotion recognition models that are used to feed a multiclass SVM that returns a final class probability. It showed a significant improvement against the baseline, confirming that the use of acquired knowledge from activity recognition is useful for group-emotion recognition and that the joint utilization of audio and video benefits the learning of the model.

Moreover, this method could benefit from OpenSMILE features in the audio-based emotion recognition module, on the development of multimodal approaches that are based on early-fusion, and on the design of a “fully” end-to-end network that receives both video and audio as input and learns the relevant features for the classification task.

Chapter 5

Interaction recognition

This chapter provides a thorough description of the main steps regarding the implementation process of the proposed solution. In the following sections, details will be provided concerning the dataset used for training, validating and testing. The performance evaluation process is also described in this chapter along with the details on the implementation of the framework. Finally, the main results are also presented. The chapter is divided in the following sections:

- **Internal Datasets**: several statistics regarding Bosch's internal dataset in terms of classes and different data splits, section 5.1.
- **Framework implementation**: details regarding the main steps of the framework implementation, particularly for the data preparation phase, section 5.2.
- **Violence detection**: details regarding the network architectures used for the task of violence detection as well as class imbalance countermeasures, section 5.3.
- **Multi-class classification**: details regarding class imbalance countermeasures and the grouping of similar classes for multi-class classification, section 5.4.
- **Evaluation**: description of the evaluation toolbox provided by Bosch and of the visualization tool that was created, section 5.5 .
- **Results**: presentation of the results of the main results regarding the violence detection task and the interaction recognition task, section 5.7.

5.1 Internal datasets

The dataset used for training, validating and testing the proposed algorithm was provided by Bosch Car Multimedia. There are a number of internal datasets available at Bosch that can be used for activity recognition purposes, however, this dissertation focused mainly on the Sunnyvale dataset. This dataset is composed of 295 videos with 9 different actor pairs performing different activities in the backseat of a vehicle. It is important to note that each video has only one pair of actors present on the backseat of the car, which is always stopped. Since there are no videos featuring a single actor, the labeled actions are related to human interactions only. Nevertheless, this dataset is mostly used as proof of concept as it is quite limited, with only nine actor pairs, all recorded under very similar conditions.

Each video file is associated with an *xml* file that includes the timestamps for every activity happening in the video. Table 5.1 summarizes some statistics regarding the actions present in this dataset. From table observation, we can see that there are 10 different interactions and that the action Talking takes up the most amount of time. We can also observe a big difference regarding the mean time of several actions, for example, between the Slapping and Talking action. This poses an additional difficulty as the model should be able to detect fast actions such as Slapping, Pushing or Punching, but also longer actions like Fighting or Talking. Therefore, this difference in mean times must be considered when selecting the number of frames in a sequence to be served as input to the model.

Table 5.1: Action statistics

Action	Fighting	Punching	Pushing	Grabbing	Slapping	Kicking	Talking	Hugging	Touching	Arguing
Time (sec)	5913	908	1363	4090	1040	574	18371	1993	1090	621
#actions/class	325	271	694	501	738	195	511	337	126	20
Mean time	18,46	3,4	1,96	8,133	1,43	2,95	36,1	5,93	8,67	31,06

5.1.1 Data splits

The next step was to determine which videos would be used for training, validating and testing the model. To evaluate the generalization capability of the model, it is crucial to have different actors in the test set and the training set. Thus, one of the nine actor pairs was selected for testing and a different actor pair was chosen for validation purposes. The remaining actor pairs were used for training the model. Table 5.2 presents the distribution of actions in each set.

There are several actions that are quite similar to each other, like the Talking and Arguing actions or the Touching and Grabbing actions. Thus, the model will probably have a hard time distinguishing between these actions. Having this in mind, the grouping of similar classes should be explored for an improved model performance. Additionally, these actions can be further divided into violent and non-violent actions to solve the problem of violence detection. Finally, we can also observe a significant imbalance among the classes in the training set which will probably affect the model's performance regarding the minority classes, as mentioned in section 2.2.5.2.

Table 5.2: Action distribution in train, validation and test sets

Action	Train set	Validation set	Test set
Arguing	8	4	8
Hugging	240	22	45
Talking	221	222	191
Touching	43	5	74
Unlabeled	27	5	1
Fighting	280	10	28
Grabbing	433	12	43
Punching	198	14	21
Pushing	399	13	232
Kicking	148	18	23
Slapping	559	21	107

5.2 Framework implementation

This section describes the implementation of the proposed framework for in-vehicle action recognition, particularly for the data preparation phase. This framework was devised in a modular way, so that it would work with different datasets or models. In this way, one can easily switch between datasets or try different models without changing the scripts responsible for loading the data. A block diagram of the data preparation phase is illustrated in figure 5.1.

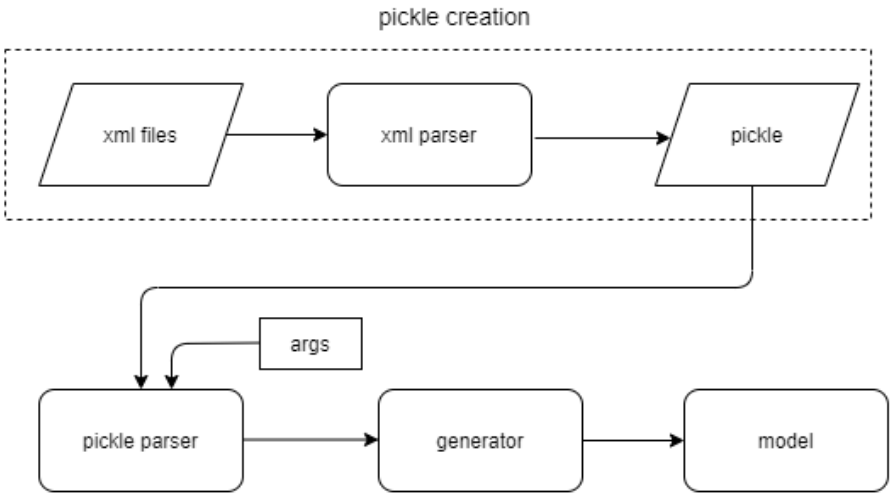


Figure 5.1: Block diagram of the data preparation phase

5.2.1 Data preparation

As the labels for each video were stored in *xml* files, the first step was to create a script to load the labels from the *xml* files and store them in a pickle. Afterwards, another script was made that reads the resulting pickle file and creates a list with information regarding sequences of an arbitrary number of frames. This list will be used by the custom generator that is responsible for fetching the frame sequences while training the model.

5.2.1.1 Xml parser

The *xml parser* is responsible for loading the labels from the respective xml file and storing them in a pickle. It also handles the conversion of the start and end time of each action to frame start and frame end.

5.2.1.2 Pickle parser

The *pickle parser* will read the resulting pickle file from the *xml parser* and convert it into a list that will be used to create different sequences of frames. First, it will split each video into different events, where each event contains only one action label, discarding events with less than min_frames. Afterwards, it will create sequences of frames by applying a sliding window with stride 8 and a window length 16 across all the frames of each event. It also handles the conversion of action labels to binary labels if necessary and includes them in the list.

5.2.1.3 Custom generator

A custom generator was created in *keras* to store the data sequentially in RAM instead of storing it in its entirety. This is especially important when dealing with large datasets and different datasets at the same time, where loading the whole dataset would lead to memory problems. This generator also performs real-time frame extraction from the video dataset when generating each batch. Thus, the training frames don't need to be extracted and stored beforehand. Finally, it also handles the pre-processing of frames, such as image normalization by mean subtraction, and performs data augmentation if needed.

5.3 Violence detection

Instead of tackling the problem of interaction recognition right from the start, an initial approach was followed in order to solve the simpler binary task of violence detection. There were no results available concerning interaction recognition using this dataset, only results related to violence detection. Therefore, these results can be used as a baseline for comparison with the performance of the implemented action recognition algorithm on this binary task.

To solve this problem, the actions in the dataset must be further divided into **violent** and **non-violent** categories, as shown in table 5.3. The Arguing action was not considered a violent

action although it can be viewed as a more aggressive form of the Talking action. As these two actions are quite similar, it is very hard to distinguish between them using only image data (RGB frames). Audio information would be a better option to differentiate between these classes as people arguing tend to speak louder or in a more aggressive fashion. The Unlabeled class was also used for training and was considered non-violent. This class includes actions that don't belong to any of the other classes but depict common behaviours of people inside vehicles, like opening and closing doors, fastening seat belts, using a mobile phone and many more.

As this is a binary classification problem, the violent class will be regarded as the **positive** class from this point forward and the non-violent class will be referred to as the **negative** class.

Table 5.3: Aggregated classes

Action	Non-violent	Violent
Arguing	✓	
Hugging	✓	
Talking	✓	
Touching	✓	
Unlabeled	✓	
Fighting		✓
Grabbing		✓
Punching		✓
Pushing		✓
Kicking		✓
Slapping		✓

5.3.1 Network architecture

The next step was to choose an adequate network architecture, capable of dealing with the topic of action recognition. There are several architectures that have been proven successful for action recognition in video, however, in very different contexts. This dissertation focused mainly on two different network architectures: the LRCN architecture described in 2.3.2.1 and the I3D network mentioned in 2.3.2.3.

5.3.1.1 LRCN

The Long-term Recurrent Convolutional Network architecture, proposed in [22], combines a CNN, which serves as a deep hierarchical visual feature extractor, with an LSTM, who is responsible for capturing temporal ordering and long-term dependencies. They instantiate their proposed architecture for three specific tasks: video activity recognition, image caption generation, and video description.

In the case of video activity recognition, the CNN base model used was a hybrid of the *CaffeNet* reference model [59], which is a slight variation of the *AlexNet* proposed in [60]. They used the UCF101 dataset, described in 2.3.1, to evaluate their proposed model. During training, the videos were resized to 240 × 320 and 227x227 crops and mirroring techniques were used in order to augment the training data. Additionally, they extract 16 frames from each video clip to be used for training. At test time, 16 frame clips with a stride of 8 are extracted from each video and the label probabilities are averaged across all frames to produce a single label prediction for an entire video clip.

The major changes regarding the implementation of this architecture for violence detection include the following: the CNN base model that was used as a feature extractor was the VGG16 proposed in [61], where the last 3 fully-connected layers were removed, as shown in figure 5.2. This architecture was chosen as it obtained a very good result in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC- 2014) classification and location tasks. A pre-trained version of this model was used, which allows us to take advantage of useful learned features by re-using the ImageNet pre-trained weights. Afterwards, a Flatten layer was added along with an LSTM layer with 128 units to capture temporal information. Finally, a dense layer with the softmax activation function is placed after the LSTM layer for classification purposes. A block diagram of the implementation of the LRCN network can be observed in figure 5.3.

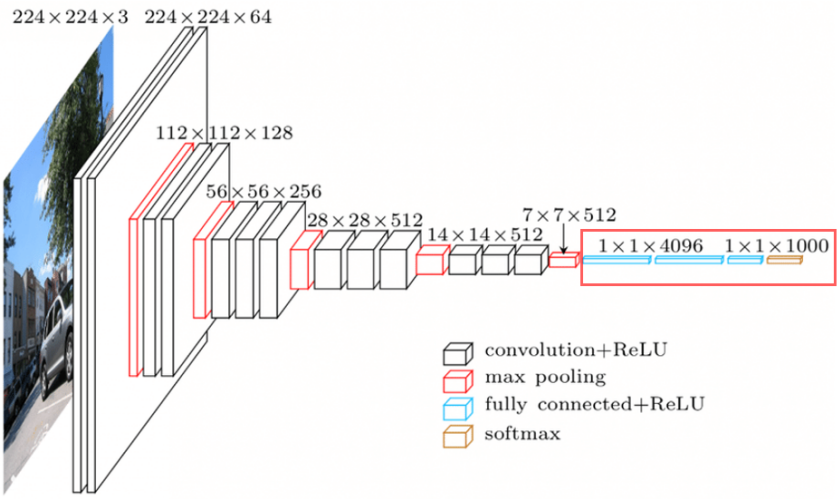


Figure 5.2: VGG16 model architecture adapted from [62], where the highlighted box corresponds to the layers that were removed.

The VGG16 is only capable of handling one image at a time, transforming it into an internal matrix or a vector representation. However, the CNN model should be applied to each input image in order to pass the output of each input image to the LSTM as a single time step. This can be accomplished with the help of a TimeDistributed layer. A **TimeDistributed layer** is able to apply the same transformation on a list of input data. Thus, the desired effect of providing a sequence of

image features to the LSTM can be achieved by wrapping the entire CNN model (in this case, the VGG16) in a TimeDistributed layer, as shown in figure 5.3.

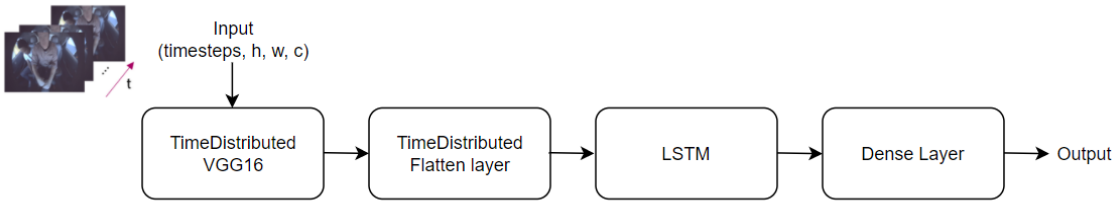


Figure 5.3: Block diagram of the implementation of the LRCN architecture.

5.3.1.2 Inflated Inception

The I3D model is a type of 3D convolutional network that was introduced in [35]. Their proposed architecture starts with a 2D convolutional architecture, the InceptionV1 proposed in [36], and inflates all the pooling and kernels to 3D. Therefore, it uses 3D convolutions in order to learn spatiotemporal features directly from videos. However, if the kernels are simply expanded to 3D, the model will have a lot more parameters which makes it harder to train. Hence, the authors also take advantage of the pre-trained parameters from 2D architectures trained on very large datasets, such as the ImageNet dataset, by bootstrapping 3D filters from 2D filters. They also report a considerable improvement in performance in standard action recognition datasets, such as UCF-101 and HMDB-51 mentioned in 2.3.1, after pre-training their models on the Kinetics dataset. Thus, a big contribution of their paper was the possibility of using transfer learning from the Kinetics dataset to other datasets for similar tasks.

The major changes regarding the replicated framework lie on the data preparation phase and the training phase. The authors trained their models on the Kinetics dataset, mentioned in section 2.3.1, using the standard SGD optimizer with a momentum value of 0,9. They used random cropping during training, by randomly cropping a 224x224 patch after resizing the video's smaller side to 256 pixels and also random flipping in order to augment the training data. Furthermore, their model was trained using 64-frame snippets while the testing was done using the whole videos and averaging predictions temporally. Figure 5.4 illustrates the Inflated Inception network architecture introduced in [35].

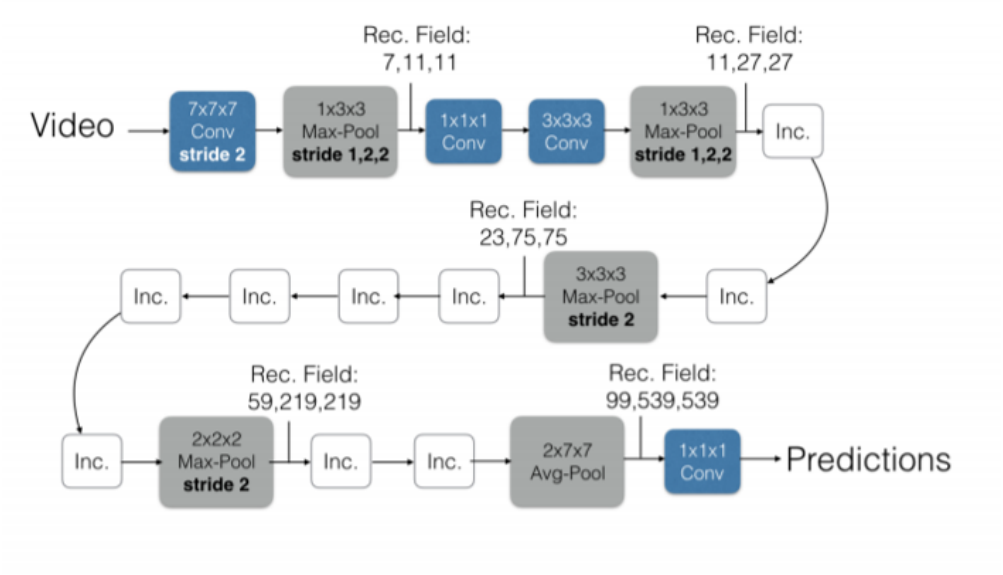


Figure 5.4: Inflated Inception network architecture, extracted from [35].

5.3.2 Class imbalance

In the case of violence detection, we can observe a significant imbalance between the positive class and the negative class after dividing the actions in the dataset into violent and non-violent categories. In fact, the positive class only represents around 20% of the training set while the negative class represents the remaining 80%.

At first, no techniques were used to deal with this class imbalance. Initial results showed a high accuracy on the test set but performed poorly in terms of the minority class, which is, in this case, the positive class. As the positive class is also the one we are trying to detect in the first place, these results were not acceptable. Therefore, several different approaches for handling class imbalance were tested and compared in order to improve the classifier's predictive performance regarding the minority class:

- **Undersampling**: An undersampling of the majority class can be performed in order to balance the training set. In this case, an undersampling of the "No violence" class was done by randomly removing samples belonging to this class until the training set was balanced.
- **Class weights**: The training algorithm can be modified to take into account the skewed class distribution by assigning different weights for each class. These weights are used in order to up-weight misclassification errors in the loss function regarding the less frequent classes. Thus, class weights that reflect the class distribution in the training set were used.
- **Focal loss**: This type of loss was originally introduced for Dense Object Detection in [63]. It is an alternative to standard cross entropy loss that down-weights the loss assigned to well-classified examples. Hence, it focuses training on a sparse set of hard examples and prevents the vast number of easy negatives from overwhelming the detector during training.

5.4 Multi-class classification

In the case of multi-class classification only the Inflated Inception network was used as it showed superior results in the previous task of violence detection. The data preparation phase is very similar for multi-class classification and uses the same scripts described in section 5.2.1. However, the Unlabeled class is removed from the dataset and only single-label frames were used for training.

In terms of class imbalance, we can also observe a significant imbalance among the different actions in the dataset. Thus, class weights that reflect the dataset's class distribution were used in order to tackle this class imbalance. An alternative to using class weights could be to undersample the majority classes, which provided slightly better results in the case of violence detection. However, as this dataset is already quite small, especially without the Unlabeled class, the use of class weights was preferred in order maintain the size of the dataset or else it would be drastically reduced.

Initially, the actions in the dataset were split into 4 different categories: Non-Violent, Violent, Kicking and Hugging. The Kicking and Hugging were separated from the Violent and Non-Violent categories, respectively, as they were also the ones that were harder to predict, resulting in a higher number of misclassifications, in the case of violence detection.

Afterwards, all the different classes were used to train the model asides from the Unlabeled class. Nonetheless, as several interactions are quite similar to one another, this might not be the best approach in order to solve the problem of multi-class classification. For example, the Arguing class is very similar to the Talking class and the Touching class is also very similar to the Grabbing class. In fact, through visual inspection of some videos, several cases were encountered in which it was very hard to distinguish between these classes. Thus, the grouping of related classes should be explored in order to improve the classifier's performance.

5.5 Evaluation

Standard precision and recall values are insufficient to describe the natures of the errors in the dataset. Event-based metrics are necessary for a more detailed picture of the performance of the model.

5.5.1 Evaluation toolbox

The evaluation toolbox provided by Bosch outputs a **frame analysis** as well as an **event analysis** for a binary classification task. However, it is not yet capable of handling multi-class evaluation. Therefore, this toolbox will only be used for the violence detection task.

The frame analysis depends on the correspondence between the ground truth label and the predicted label. On the other hand, the event analysis checks if each event is detected, whether the event is detected at the correct time, and how closely the predicted event boundaries correspond to the true start and stop times.

5.5.1.1 Error categorization

The deviation between the prediction and the ground truth label can happen for several reasons. This toolbox categorizes the possible errors in the following way:

- **Occurrence errors**: Insertion and Deletion.
- **Timing errors**: Overfill and Underfill.
- **Segmentation errors**: Merging and Fragmentation.

These types of errors are further illustrated in figures 5.5, 5.7 and 5.6.

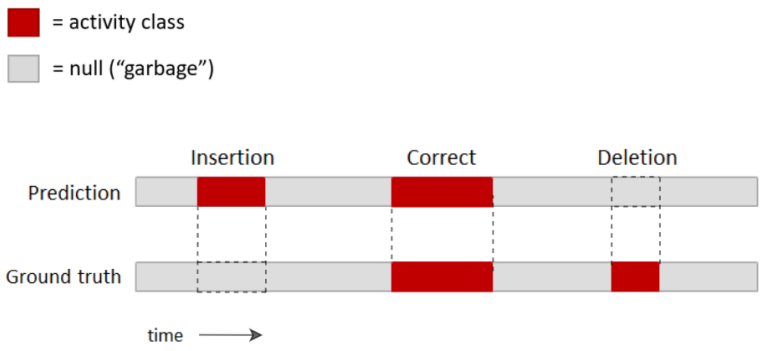


Figure 5.5: Occurrence errors

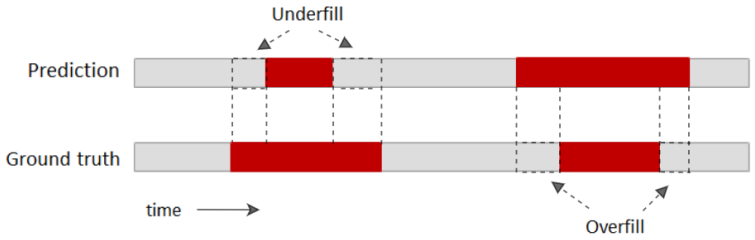


Figure 5.6: Timing errors

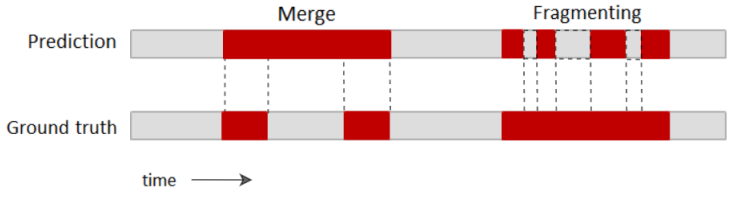
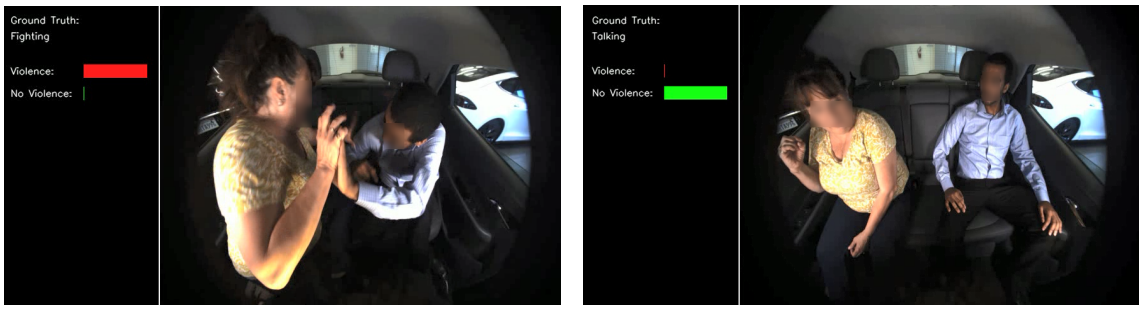


Figure 5.7: Segmentation errors

In the case of the event-based analysis, a detection is considered correct if the action track that it belongs to was correctly classified and the intersection-over-union ratio of the detection and ground truth bounding box is greater than a specific threshold.

5.5.2 Visualization tool

A visualization tool was created using the OpenCV library in order to visualize the results obtained by the model. This tool creates a confidence bar alongside each video that matches the confidence of the prediction of the model for each frame. Above the confidence bar, the ground truth labels are also presented to check if the video is correctly labeled. This tool was very useful for visualizing the results and understanding why and in which cases the model was failing the most.



(a) Fighting scene from test set video.

(b) Talking scene from test set video.

Figure 5.8: Visualization tool applied to a test set video.

5.6 Model optimization

The training phase of a deep learning model can be a long one, especially when the model has a high number of parameters and is fairly complex. In the case of the LRCN architecture, the bottleneck in terms of training is the LSTM layer as it requires a sequential processing, not taking advantage of the parallel structure of the GPU hardware it was trained on. On the other hand, the Inflated Inception network is a type of 3D convolutional network, which is a deep and complex network with a high number of parameters. However, by re-using pre-trained weights trained on the Kinetics and ImageNet datasets, the convergence time is greatly reduced.

In the case of the LRCN network architecture, the LSTM layer along with the dense layer that was included were randomly initialized, yet, the convolutional layers of the VGG16 have already learned rich, discriminative features. If the gradient is allowed to backpropagate from these random values all the through the network, these powerful features might be destroyed. To avoid this, only the fully-connected layer along with the LSTM layer were trained while the weights of the pre-trained VGG16 were, initially, frozen. Afterwards, the weights of the CNN were unfrozen and the training was resumed with a reduced learning rate.

As previously mentioned in section 5.1.1, one actor pair was selected for the test set (actor pair A5A6) and another one for the validation set (actor pair A3A4). However, results from both architectures show a big overfitting quite early on, as the validation loss starts to increase almost right away while the training loss keeps decreasing steadily for several epochs. As this dataset is quite limited with only 9 different actor pairs recorded all under very similar conditions in an indoor-environment, the model might be having a hard time generalizing properly from one actor pair to another. The fact that different actors tend do perform the same action in different ways is probably also affecting the model's ability to generalize. Several regularization techniques were employed in an attempt to reduce overfitting, including the following:

- **Dropout**: This technique results in randomly selected neurons being ignored ("dropped") during training. Hence, it has the effect of reducing the capacity or thinning the network while training, due to the outputs of a layer under dropout being randomly sub-sampled [64]. Thus, a dropout layer was added to the model and different values were experimented with (20%, 50%, 70%).

- **Data augmentation**: This technique relies on artificially creating new training data from existing training data. It is a widely used technique as it is responsible for increasing the diversity of the data, which, in turn, leads to a better generalization capacity. In this case, random horizontal flips and small translations were performed with the help of the *imgaug* library in order to augment the training data,.

As these regularization techniques were not successful in preventing this early overfitting, a new validation set was created using the same actor pairs that were included in the training set. To accomplish this, the previous validation set was grouped together with the training set and a new validation set was created from this group, resulting in a 80/20% train/validation split. With this new validation set, we can check if the model still overfits when using the same actor pairs in both the training set and the validation set. In fact, with this new validation set, we were able to prevent the model from overfitting and the performance was greatly improved in terms of validation metrics.

In terms of optimization algorithms, several optimizers were used, including the following:

- **Stochastic Gradient Descent (SGD)**: This optimizer is a variation of the Gradient Descent optimizer, in which the model parameters are updated on every iteration. These frequent updates lead to a faster convergence at the cost of an increased parameter variance. In contrast, the Gradient Descent algorithm updates the weights after calculating the gradient on the whole dataset, which results in a slower process that requires a large amount of memory.

- **Adaptive Moment Estimation (Adam)**: The Adam optimizer is a very popular choice in the field of deep learning that is known for achieving good results fast. It computes adaptive learning rates for each parameter from estimates of first and second moments of the gradients [65].

- **Nesterov Adaptive Moment Estimation (Nadam)**: This optimizer has a similar implementation to the Adam optimizer, but replaces the momentum component with Nesterov momentum. “Nesterov momentum is theoretically and often empirically superior to momentum.” [66].

"Insofar, RMSprop, Adadelta, and Adam are very similar algorithms that do well in similar circumstances. [...] its bias-correction helps Adam slightly outperform RMSprop towards the end of optimization as gradients become sparser. Insofar, Adam might be the best overall choice." [67].

5.6.1 Hyper-parameter tuning

In machine learning, **hyper-parameters** are variables or parameters that are external to the model and are, therefore, set before training. On the other hand, model parameters (for example, model weights) are variables that are internal to the model and are derived via training.

Hyper-parameters are very important as they will determine how the learning process unfolds, which will impact the performance of the model under training. They include variables that determine the network structure, such as the number of hidden units, and variables that control how the network is trained (for example, the learning rate). Hence, selecting appropriate hyper-parameters is key due to their impact on the performance of the model.

In this case, the SGD (with a momentum value of 0.9), Adam and Nadam optimizers were tested with different learning rates (1e-4, 1e-5 and 1e-6). The batch size, which is related to the number of training samples that are propagated through the network, was also altered as it also affects training time and performance. In this case, a batch size of 16 and 32 were tested. However, as we are dealing with video data and each sample corresponds to a sequence of frames, a batch size of 32 caused some memory issues and, therefore, a batch size of 16 was preferred.

Another important parameter that needs to be carefully selected as it will greatly impact the performance of the model is related to the number of frames that should be included in each sequence. This I3D network was originally trained with 64-frame snippets, which corresponded to a temporal footprint of 2,56 seconds. However, as shown in table 5.2, the Pushing and Slapping interactions are quite fast and have a significantly lower mean time than other interactions. For example, the Slapping action has a mean time of only 1,43 seconds which is equivalent to approximately 35 frames of the video, as these were all recorded at 25 frames per second (FPS) . Thus, 64 frames would be inappropriate for the model to be able to detect these fast actions and so, 16-frame sequences and 32-frame sequences were tested and compared. In fact, results show that the model is worse at detecting the Slapping and Pushing actions as violent actions when using 32 frames per sequence.

5.7 Results

5.7.1 Violence detection

The results displayed in the following section are related to the binary classification problem of violence detection between passengers inside a vehicle. The evaluation toolbox, mentioned in section 5.5.1, was also used in order to evaluate and compare the performance of the models. The results obtained from this detailed evaluation can be consulted in Annex A.

5.7.1.1 LRCN architecture

As mentioned in section 5.6, this network was initially trained with different actor pairs from the one used in the validation set. However, results show a big overfitting early on which indicates that the model is not being able to generalize properly from one actor pair to another. Therefore, a new validation set was created from the same actor pairs that were included in the training set to confirm this generalization problem. In figures 5.9 and 5.10, we can observe how the loss, accuracy, precision and recall values progress, using this new validation set and the Adam optimizer with a learning rate of 1e-5.

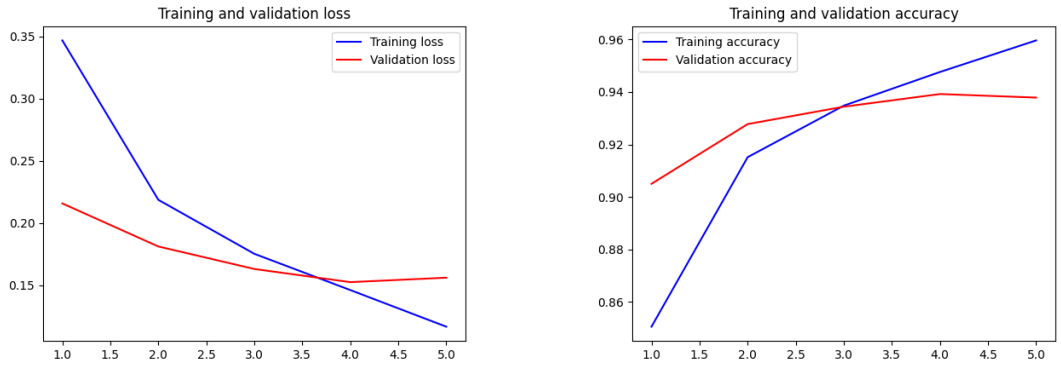


Figure 5.9: Epoch progression of loss and accuracy values, optimized by the Adam algorithm.

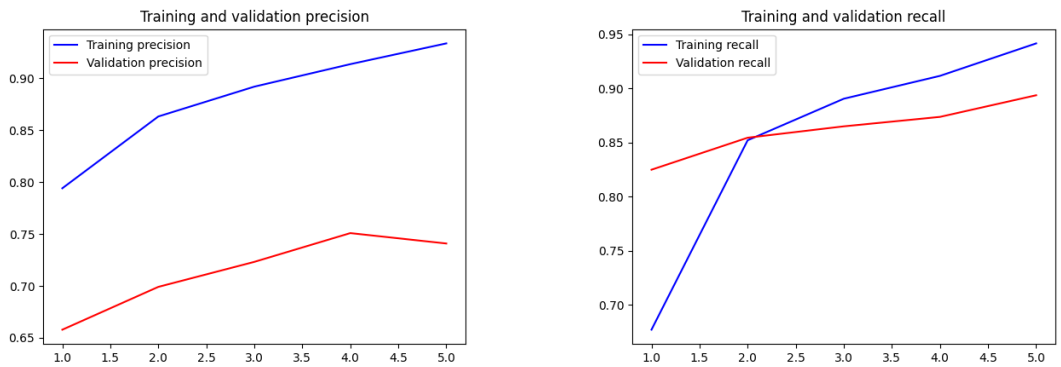


Figure 5.10: Epoch progression of precision and recall values, optimized by the Adam algorithm.

The results obtained using this new validation set were greatly improved in terms of validation metrics, indicating that the model was in fact having trouble generalizing from a set of actor pairs to one it had not seen before. In terms of results regarding the test set, this network obtained an accuracy of 83%. However, as mentioned in section 2.2.5.1, this metric is sensitive to imbalanced data and is, therefore, not an appropriate metric for evaluating the performance of the model. Table 5.4 presents several additional metrics that are usually used for evaluating the performance of a classifier. Additionally, the confusion matrix is also presented in table 5.5.

Table 5.4: Precision, recall and F_1 score values regarding the test set for the LRCN network.

Class	Precision	Recall	F_1 score
No Violence	0.85	0.95	0.90
Violence	0.80	0.48	0.60

Table 5.5: Confusion matrix for the test set regarding the LRCN network.

		Predicted values	
		Positive (P)	Negative (N)
Actual values	Positive (P)	TP 28776	FN 30627
	Negative (N)	FP 6982	TN 154536

The false positives that were obtained were mainly related to the Hugging and Touching actions. On the other hand, the false negatives produced were mainly in fast actions such as Slapping, Pushing or Punching and also in several Grabbing scenes.

5.7.1.2 Inflated Inception network architecture

The Inflated Inception network was able to achieve superior results compared with the LRCN network architecture. This network was first trained using the SGD optimizer with a momentum value of 0.9, as it was also the one used in [35], where the I3D network was introduced. Afterwards, the Adam and Nadam optimizers were used. The results obtained using the Adam optimizer were very similar to those obtained using the Nadam optimizer. However, the Adam optimizer resulted in a slightly higher performance regarding the F_1 score metric. On the other hand, the SGD optimizer with Nesterov momentum yielded inferior results compared with these two optimizers. These experiments show that the SGD optmizer might not be the best option for finding the local minima during training on this dataset. Instead, adaptive learning rate optimizers with momentum demonstrated an increased performance, and have the added advantage of faster convergence. In figures 5.11 and 5.12, we can observe how the loss, accuracy, precision and recall values progress for each epoch during the training phase using the Adam optimizer with a learning rate of 1e-6 and sequences of 16 frames.

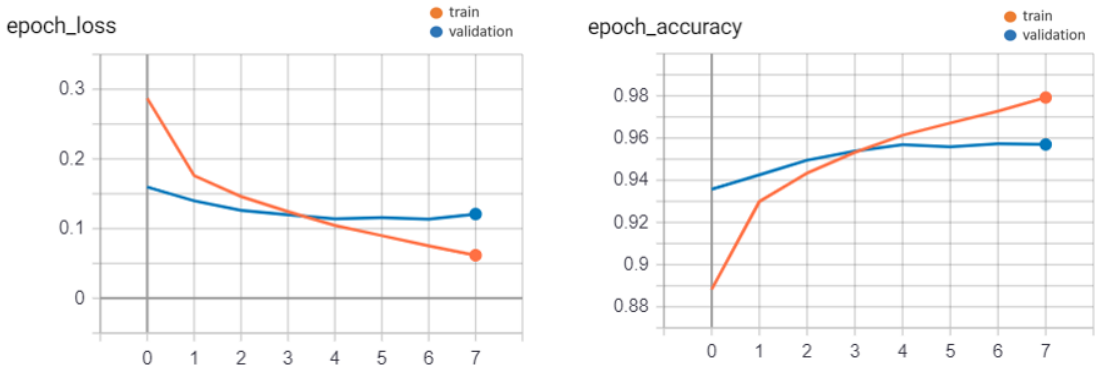


Figure 5.11: Epoch progression of loss and accuracy values, optimized by the Adam algorithm.

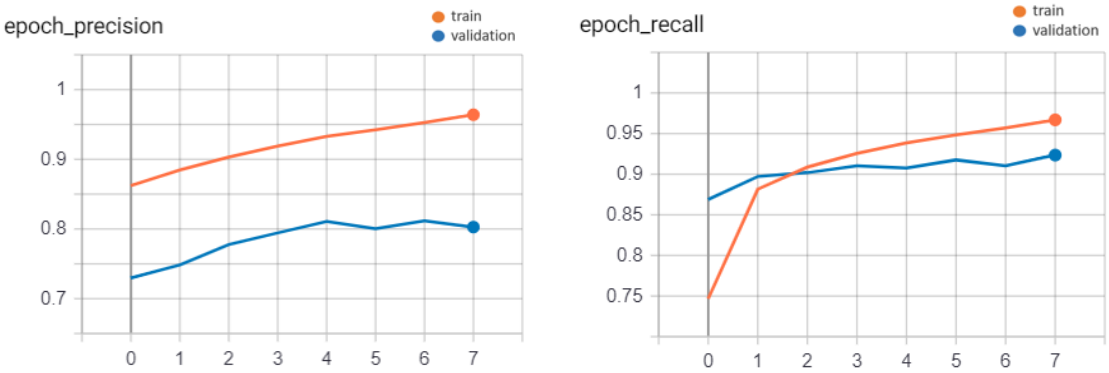


Figure 5.12: Epoch progression of precision and recall values, optimized by the Adam algorithm.

The best results were achieved with a balanced version of the training set, by under-sampling the majority class, using the standard cross-entropy loss. When using class weights to tackle the class imbalance problem, results were relatively similar but were able to achieve a higher recall value on the test test, at the cost of a decrease in precision. However, the gain in recall when using class weights, is not worth the trade-off in precision. When using the focal loss function instead of the standard cross-entropy loss results were also similar but slightly worse than the ones obtained when using a balanced version of the training set, by under-sampling the majority class.

In terms of results regarding the test set, the I3D network was able to achieve an accuracy of 83%. Moreover, the Precision, Recall and F1 score values are presented Table 5.6. The false positives that were produced were mainly related to the Hugging and the Touching classes. The Touching class is very similar to the Grabbing class, and some of the false positives could even be considered cases of incorrect labelling. For example, in figure 5.13, which corresponds to a false positive produced during a Touching scene, should clearly be considered a Grabbing scene. On the other hand, the false negatives were mainly on Pushing and Slapping scenes and also some Grabbing scenes. The Slapping and Pushing interactions are characterized by being very fast actions and, therefore, are harder to be correctly detected by the model. Furthermore, the Grabbing scenes that originated these false negatives could be easily confused with the Touching class, which is considered a non-violent interaction, as shown, in figures 5.14. In fact, the Grabbing interaction

was considered a violent action as many of the Grabbing scenes are somewhat aggressive, yet, through visual inspection of the dataset, several of the Grabbing scenes should certainly not be considered as Violence.

Figure 5.13: False positive produced during Touching scene from a test set video.

Figure 5.14: False negative produced during Grabbing scene from a test set video.

Table 5.6: Precision, recall and F_1 score values regarding the test set for the violence detection task.

Class	Precision	Recall	F_1 score
No Violence	0.94	0.83	0.88
Violence	0.63	0.84	0.72

To further validate the performance of the model, these results were compared to an anomaly detection framework, proposed in [4], using the same test set, described in section 5.1.1. The main goal of this anomaly detection algorithm is to detect abnormal actions (which are, in this case, violent actions). However, this framework was only trained with non-violent actions from the training set, unlike in the action recognition algorithm, which was trained on violent and non-violent actions. Moreover, the IVS anomaly detection framework outputs a regularity score, which, in this case, corresponds to a non-violent score, where lower values represent predictions for violent frames while higher values represent predictions for non-violent frames. A comparison of the true positive, false positive, true negative and false negative rates of both algorithms is presented in table 5.7. Additionally, the confusion matrix regarding the test set was also computed for both frameworks, as shown in figures 5.8 and 5.9.

Table 5.7: True positive rate (TPR), False negative rate (FNR), True negative rate (TNR) and False. positive rate (FPR) of the anomaly detection and action recognition algorithms.

Algorithm	TPR	FNR	TNR	FPR
Anomaly detection	0.79	0.21	0.67	0.33
Action recognition	0.84	0.16	0.83	0.17

Table 5.8: Confusion matrix for the action recognition algorithm regarding the test set.

		Predicted values	
		Positive (P)	Negative (N)
Actual values	Positive (P)	TP 49371	FN 28442
	Negative (N)	FP 9147	TN 133961

Table 5.9: Confusion matrix for the anomaly detection algorithm regarding the test set.

		Predicted values	
		Positive (P)	Negative (N)
Actual values	Positive (P)	TP 47671	FN 13033
	Negative (N)	FP 52872	TN 107380

After comparing results from both algorithms, the difference in performance is clear. The action recognition algorithm has an overall better performance concerning all metrics displayed in table 5.7, with a higher true positive rate and a significantly lower false positive rate. Thus, the action recognition algorithm proved to more effective in correctly identifying violent actions.

Nevertheless, the action recognition framework uses a supervised learning algorithm, meaning that each example in the training set has to be associated with a label. This can be a problem as there is an overwhelming variety of human behaviours. Furthermore, there might not be enough training data available regarding some interactions that rarely happen. On the other hand, the anomaly detection framework uses a semi-supervised learning algorithm which only requires the definition of the expected normal data and, hence, can be used on datasets with reduced labeling.

The I3D model was subsequently trained with **motion features** instead of image data to see if it would lead to a better generalization capacity. To accomplish this, the TV-L1 algorithm [68] was used to compute the optical flow for each video of the dataset beforehand. The Precision, Recall and F_1-score values obtained for the test set can be observed in table 5.10.

Table 5.10: Precision, recall and F_1 score values regarding the test set using optical flow as input.

Class	Precision	Recall	F_1 score
No Violence	0.95	0.80	0.87
Violence	0.66	0.85	0.74

The results obtained when using optical flow instead of RGB frames showed a slight improvement in terms of Precision (from 63% to 66%) and a minor increase regarding the Recall value (from 84% to 85%), as shown in table 5.12. However, it is important to note that this dataset only includes videos recorded in an indoor environment with the car stopped. The contribution of optical flow will most likely be worse when using videos where the car is moving due to camera trepidation. Moreover, optical flow algorithms usually perform worse when using a Near-Infrared (NIR) camera which is necessary to record the vehicle's interior in low illumination settings (e.g. at night).

Finally, a two-stream approach using both optical flow and RGB data was tested as these types of architectures have exhibited an improved performance in the past when compared with unimodal architectures. Table 5.11 displays several metrics regarding the performance of this approach on the test set. Additionally, a comparison of the performance of the model using only RGB frames, optical flow or both RGB and flow as input is presented in table 5.12.

Table 5.11: Precision, recall and F_1 score values regarding the test set using a 2-stream approach.

Class	Precision	Recall	F_1 score
No Violence	0.94	0.83	0.88
Violence	0.67	0.84	0.74

Table 5.12: Performance on test set regarding the Violence class using RGB, flow or RGB+flow as input to the model.

Input	Precision	Recall	F_1 score
RGB	0.63	0.84	0.72
Optical Flow	0.66	0.85	0.74
RGB + Flow	0.67	0.84	0.74

The results displayed in table 5.12 revealed that the flow-based model exhibited an increased performance compared with the RGB-based model and a similar performance compared with the two-stream approach. Therefore, these results indicate that the contribution from flow alone is slightly higher than that of RGB for this particular task. Nevertheless, despite RGB-only detection showing a higher generalization problem, it has the advantage of being a lighter approach compared with the other two approaches.

5.7.2 Multi-class classification

The results presented in the following section are related to the task of interaction recognition between passengers inside a vehicle. As mentioned in section 5.4, the interactions were first divided into 4 categories: Violence, No violence, Kicking and Hugging. As the Kicking and Hugging classes initially resulted in a higher error rate for the violence detection task, these were separated to see if it would lead to a increased performance. Table 5.13 shows the results obtained with the I3D network, regarding the test set.

Table 5.13

Class	Precision	Recall	F1 score
No Violence	0.97	0.37	0.54
Violence	0.47	0.34	0.40
Kicking	0.50	0.66	0.55
Hugging	0.34	0.8	0.48

Afterwards, the Inflated Inception network was trained with 10 classes, where each class represents a different interaction of the dataset. Initial results showed that the model was not able to differentiate between the Talking and Arguing classes, which was to be expected as these are especially hard to distinguish using only image data and no audio information. Thus, the Arguing and the Talking classes were grouped together due to the similarities between them. In figure 5.15 we can observe the epoch progression of the loss and accuracy values, using the Adam optimizer with a learning rate of 1e-5. In order to evaluate the performance of the model on the test set, a classification report providing the Precision, Recall and F_1 score values for each class was generated, with the help of the *sklearn* library, as shown in table 5.14. Moreover, the confusion matrix for the test set was also computed and is displayed in table 5.15.

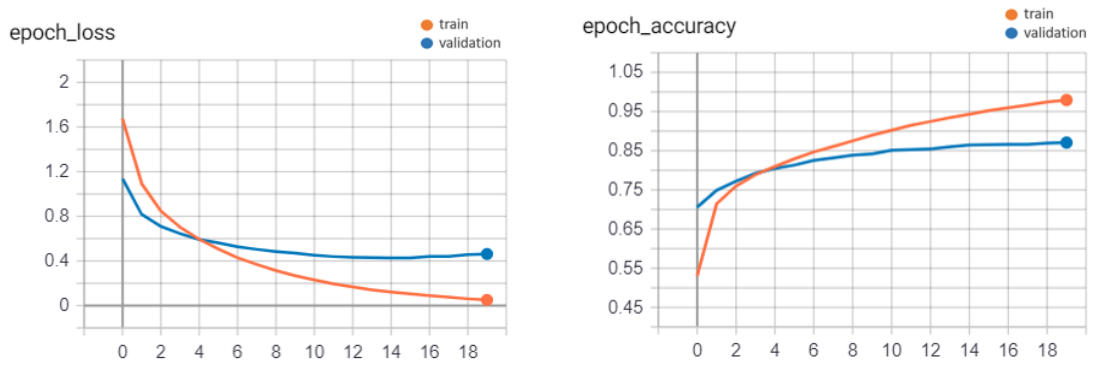


Figure 5.15: Epoch progression of loss and accuracy values, optimized by the Adam algorithm.

Table 5.14: Classification report regarding the test set, provided by the *sklearn* library.

Class	Precision	Recall	F_1 score
Talking/Arguing	0.95	0.79	0.86
Hugging	0.36	0.72	0.48
Touching	0.00	0.00	0.00
Grabbing	0.13	0.31	0.18
Pushing	0.17	0.21	0.18
Punching	0.64	0.15	0.24
Fighting	0.29	0.59	0.39
Kicking	0.38	0.69	0.49
Slapping	0.24	0.14	0.17

The classification report displayed above revealed that the model was not able to detect the Touching class at all. In fact, the confusion matrix in table 5.15 shows that the Touching class was mostly misclassified as Grabbing, Hugging or Fighting, in descending order. As previously mentioned in section 5.7.1.2, the Touching and Grabbing classes are very similar and there are several cases of incorrect labelling regarding these two classes in the dataset. Nevertheless, the Touching and Grabbing actions were not grouped together as the former was considered a non-violent action and the latter a violent one. Therefore, the Touching class was subsequently removed from the dataset and the network was trained using only 8 classes which led to an increase in accuracy (from 61% to 66%). Additionally, the results also reveal that the model has a difficult time detecting fast actions such as Pushing, Punching and Slapping. The classification report obtained after removing the Touching class from the dataset is shown in Table 5.16.

Table 5.15: Confusion matrix for the interaction recognition task regarding the test set.

		Predicted Classes								
		Talk/Arg	**Hug**	**Touch**	**Grab**	**Push**	**Punch**	**Fight**	**Kick**	**Slap**
Actual Classes	**Talk/Arg**	82243	1921	95	7483	3584	98	1571	4953	1894
	Hug	15	3627	3	674	316	1	399	1	4
	Touch	835	2759	1	3387	1277	2	2178	183	105
	Grab	2302	847	6	2479	684	14	1453	164	124
	Push	633	500	63	1684	1541	86	1712	535	695
	Punch	44	69	2	233	278	1108	5202	159	544
	Fight	230	109	7	2676	1210	143	7790	792	324
	Kick	160	0	0	230	31	41	1553	4510	5
	Slap	424	106	14	748	408	248	5095	540	1193

Table 5.16: Classification report regarding the test set after removing the Touching class.

Class	Precision	Recall	F_1 score
Talking/Arguing	0.96	0.82	0.88
Hugging	0.37	0.77	0.50
Grabbing	0.10	0.17	0.12
Pushing	0.22	0.10	0.13
Punching	0.61	0.06	0.12
Fighting	0.31	0.79	0.44
Kicking	0.79	0.49	0.61
Slapping	0.19	0.11	0.14

Motion features were also used to train the network instead of image data in an attempt to increase the model's generalization capacity as the contribution from optical flow was slightly higher for the violence detection task. However, in this case, using optical flow instead of RGB frames led to a slight decrease in accuracy (from 66% do 60%) as the model has a harder time correctly detecting the Talking and Fighting classes, as shown in table 5.17. On the other hand, results showed an increase in performance regarding the rest of the classes, especially when it comes to faster actions such as Slapping, Pushing and Punching.

Table 5.17: Classification report regarding the test set using optical flow as input.

Class	Precision	Recall	F_1 score
Talking/Arguing	0.98	0.69	0.81
Hugging	0.34	0.83	0.48
Grabbing	0.11	0.36	0.17
Pushing	0.24	0.19	**0.21**
Punching	0.62	0.17	**0.27**
Fighting	0.32	0.67	0.43
Kicking	0.77	0.55	0.64
Slapping	0.25	0.21	**0.23**

Furthermore, a two-stream configuration was also explored. The results displayed in table 5.18 are related to the performance on the test set when using a 2-stream approach.

Table 5.18: Classification report regarding the test set using a two-stream configuration.

Class	Precision	Recall	F_1 score
Talking/Arguing	0.97	0.80	0.88
Hugging	0.39	0.87	0.54
Grabbing	0.12	0.25	0.16
Pushing	0.38	0.10	0.16
Punching	0.82	0.08	0.15
Fighting	0.31	0.82	0.45
Kicking	0.83	0.57	0.68
Slapping	0.27	0.14	0.18

The two-stream approach obtained the best results in terms of overall performance and offers a better compromise between the performance of the RGB model and the flow model. It showed a significant increase in performance in longer actions when compared with the flow model, due to the additional RGB stream, and only a slight decrease in performance regarding faster actions. However, this approach is considered heavy and computationally expensive as it includes two different streams and requires the pre-computation of optical flow. Thus, a lighter approach, such as the RGB-only detection, might be preferred as it is less computationally expensive and provides similar results to the ones obtained with the two-stream configuration.

Chapter 6

Conclusions and future work

In recent years, the area of autonomous driving has attracted a lot of interest from the scientific community and is an area that is expected to grow and thrive in the next few years or decades. With the emergence of SAV's, there has been an increasing interest in developing an adequate system to compensate for the absence of the driver. In this case, an interior analysis of the vehicle is extremely important given the lack of a human driver who can monitor and intervene to avoid potential dangerous situations and ensure the safety of the vehicle's passengers. This dissertation focused mainly on recognizing different interactions between passengers inside a vehicle.

In chapter 2 several fundamental concepts for action recognition in video are explored. Furthermore, the state-of-the-art approaches used for action recognition are described and compared. The use of convolutional networks with recurrent network elements, such as the LSTM, was popularized with the introduction of the LRCN network in [22]. Nevertheless, the best results reported in literature regarding standard action recognition datasets, like UCF-101 and HMDB-51, were achieved with inflated convolutional networks, such as the I3D presented in [35]. Furthermore, results indicate that two-stream architectures exhibit a superior performance on all landmark datasets. However, these usually rely on additional handcrafted features such as pose features or motion features (e.g. optical flow) that are typically computed beforehand. Having this in mind, two different network architectures were used for solving the problem of action recognition in video: a convolutional network with recurrent network elements and the I3D network architecture. Afterwards, a thorough analysis of the performance of both networks was performed with the goal of selecting one that could best be adapted for in-vehicle action recognition.

Chapter 4 covered the experiments that were carried out in collaboration with INESC TEC for the EmotiW competition, specifically, for the sub-challenge of "Audio-video Group Emotion Recognition". This task shares some similarities to the recognition of human action from video, which is the main focus of this dissertation. The potential advantage of using audio information, for a more robust classification, was also explored in these experiments. The results indicate that, while audio alone is not suited for emotion recognition at the group level, it offers additional information that is essential for the robustness and accuracy of the model.

Chapter 5 provides a thorough description of the proposed framework for in-vehicle action recognition and details regarding its implementation. The performance of two different network architectures was compared for the task of violence detection. Additionally, several techniques for handling the class imbalance present in the dataset were tested. The results indicate that the I3D network architecture is more appropriate for activity recognition, as it surpassed the LRCN network in terms of performance for the violence detection task.

Furthermore, the results were also compared to an anomaly detection algorithm [4], also working as a violence detector, and it was verified that the action recognition algorithm presents a higher true positive rate and and a considerably lower false positive rate, as observed in table 5.7. Thus, the proposed action recognition algorithm was able to significantly outperform the anomaly detection algorithm for the violence detection task. Nevertheless, the anomaly detection framework has the advantage of using a semi-supervised learning algorithm, which allows for the use of datasets with reduced labeling. Therefore, this framework could be used to support the action recognition algorithm in detecting unknown actions and movements.

Afterwards, the I3D network was used for the task of interaction recognition. The model was initially trained with 10 classes, where each class represents a different interaction. The Arguing and the Talking classes were subsequently grouped together due to the similarities between them, which led to significant increase in performance. Finally, the Touching action was removed from the dataset which, in turn, led to an even higher increase in performance. However, the model still exhibits a low performance when it comes to fast actions such as Slapping, Pushing or Punching. Furthermore, the model has a hard time distinguishing between the Fighting, Slapping and Punching actions. This can be explained by the fact that the Fighting action typically includes occurrences of Slapping and Punching actions. Most importantly, the model's generalization ability is affected by this limited dataset, with only nine actor pairs, recorded under very similar conditions. A more extensive dataset, including more actor pairs, recorded under different conditions and with a higher action variability would lead to an increased generalization capacity.

6.1 Future Work

In terms of future work, an interesting approach would be to use pose features instead of motion features to see if it would lead to an increased model generalization capacity. An advantage of using optical flow as input to the network, instead of the image data (in this case, sequences of RGB frames), is that optical flow is invariant to appearance. Hence, the model can focus solely on the motion of the actors in the dataset, disregarding their physical appearance. However, in the context of SAVs, optical flow might not be the most viable option for the reasons discussed in section 3.3. On the other hand, pose features allow the model to focus only on the people present in the scene. The main limitation associated with pose-based action recognition lies on the difficulty to estimate high quality poses from action videos. However, recent pose estimators usually achieve good results even under poor lighting conditions or with occluded body parts. Moreover, an alternative approach could be based on the successful two-stream architectures for

action recognition, like the one in [35], which usually use motion or pose features besides the image data.

Finally, audio information could also be used for a more robust classification, through score-level or late fusion, as done in chapter 4. In this case, audio would be beneficial to distinguish between certain similar actions such as the Talking and Arguing action as these are very, with the main difference being that the Arguing action is usually louder than the Talking action.

Appendix A

Annex A: Evaluation toolbox results

Several figures that were referenced throughout the text are presented below.

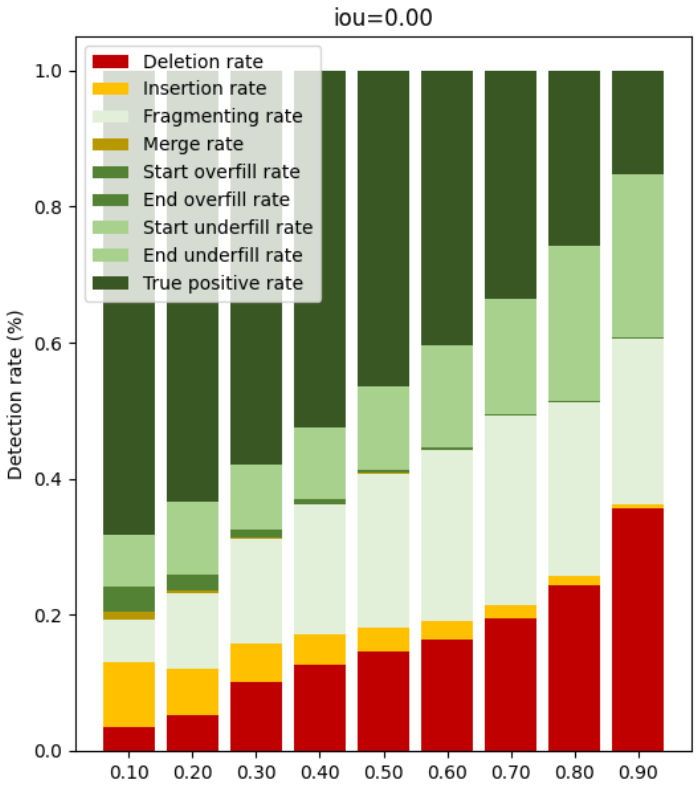


Figure A.1: Frame analysis for the violence detection algorithm, regarding the test set with the LRCN network architecture.

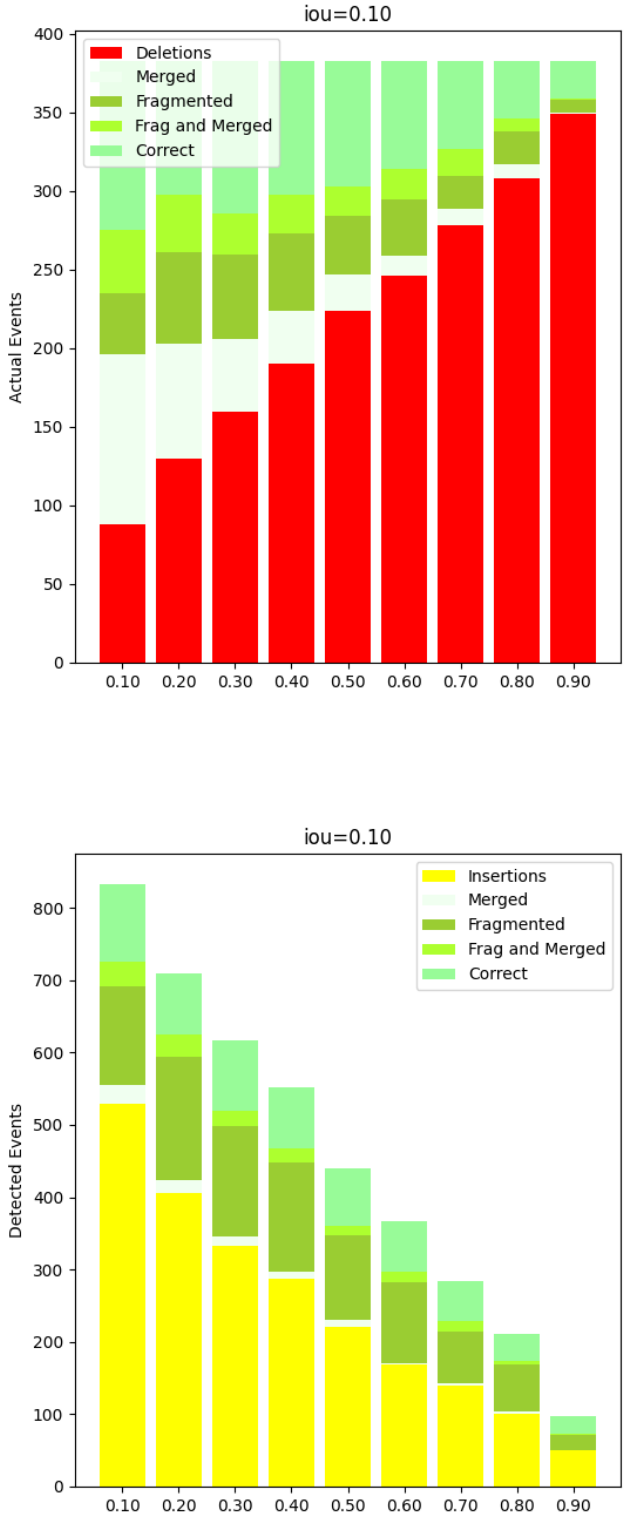


Figure A.2: Event analysis for the violence detection algorithm, regarding the test set with the LRCN network architecture.

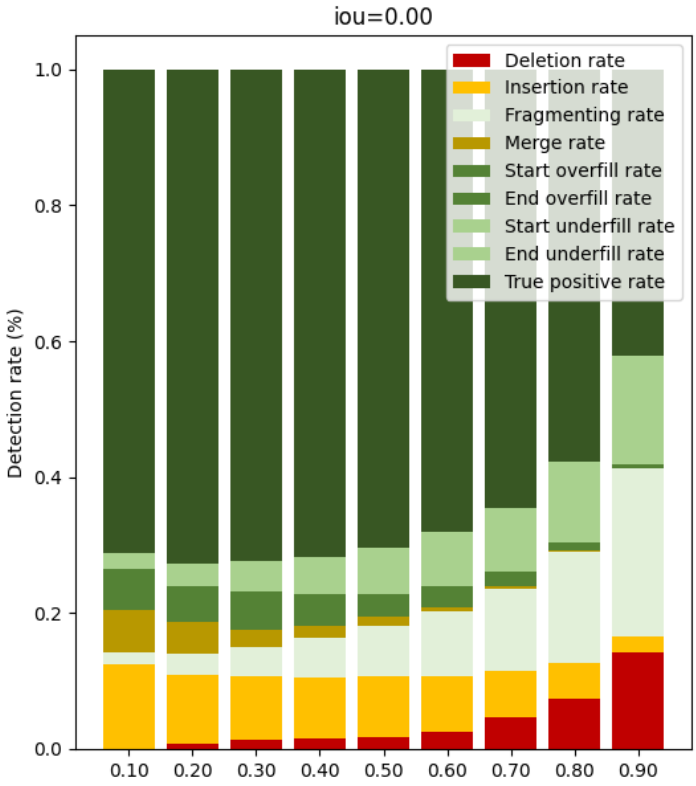


Figure A.3: Frame analysis for the violence detection algorithm, regarding the test set with the I3D network architecture.

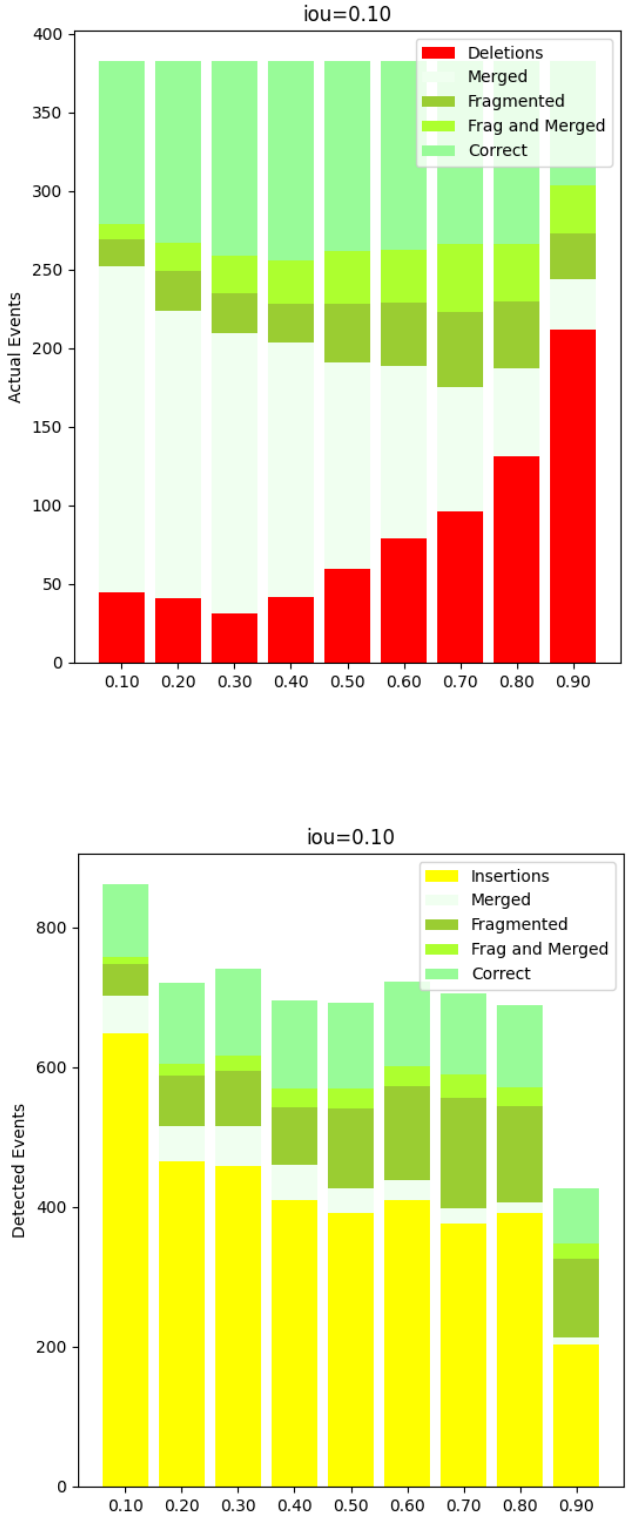


Figure A.4: Event analysis for the violence detection algorithm, regarding the test set with the I3D network architecture.

Appendix B

Annex B: Multi-class classification results

Table B.1: Confusion matrix for the interaction recognition task regarding the test set, after removing the Touching class, using RGB frames.

		Predicted Classes							
		Talk/Arg	**Hug**	**Grab**	**Push**	**Punch**	**Fight**	**Kick**	**Slap**
Actual Classes	**Talk/Arg**	85101	3870	8240	1175	83	2357	528	2477
	Hug	4	3895	456	117	0	567	1	0
	Grab	2562	1642	1388	305	9	2019	4	144
	Push	527	736	2072	691	47	2769	61	546
	Punch	2	57	100	39	492	6747	12	190
	Fight	213	135	1491	367	31	10554	225	264
	Kick	99	4	98	11	18	3052	3230	18
	Slap	190	121	698	394	128	6376	31	838

Table B.2: Confusion matrix for the interaction recognition task regarding the test set using optical flow as input.

		Predicted Classes							
		Talk/Arg	**Hug**	**Grab**	**Push**	**Punch**	**Fight**	**Kick**	**Slap**
Actual Classes	**Talk/Arg**	71929	5863	16408	2744	198	3611	459	2619
	Hug	2	4169	518	50	0	299	0	2
	Grab	1195	1488	2884	493	6	1852	1	154
	Push	183	423	2126	1443	106	2317	44	807
	Punch	8	101	857	253	1312	3851	5	1252
	Fight	60	92	2335	658	238	8887	549	461
	Kick	224	4	305	178	84	2082	3607	46
	Slap	158	204	1179	255	168	4957	49	1806

Table B.3: Confusion matrix for the interaction recognition task regarding the test set, using a two-stream approach.

		Predicted Classes							
		Talk/Arg	**Hug**	**Grab**	**Push**	**Punch**	**Fight**	**Kick**	**Slap**
Actual Classes	**Talk/Arg**	83197	4437	10376	711	47	3092	296	1675
	Hug	0	4385	338	24	0	293	0	0
	Grab	1705	1857	2036	168	0	2205	0	102
	Push	279	436	2178	766	29	3036	26	699
	Punch	0	24	210	41	635	6193	0	536
	Fight	146	76	1345	208	24	10828	417	236
	Kick	99	0	100	13	7	2564	3735	12
	Slap	169	67	854	109	37	6308	24	1208

References

[1] Susan Shaheen, Nelson Chan, Apaar Bansal, and Adam Cohen. Shared mobility: Definitions, industry developments, and early understanding. *University of California Berkeley Transportation Sustainability Research Center, Berkeley. http://innovativemobility. org/wp-content/uploads/2015/11/SharedMobility_WhitePaper_FINAL. pdf*, 2015.

[2] Mark Thomas. Reinventing carsharing as a modern (and profitable) service. In *2018 ITS America Annual Meeting Detroit*. ITSWC, 2018.

[3] Adam Stocker and Susan Shaheen. Shared automated vehicle (sav) pilots and automated vehicle policy in the us: Current and future developments. In *Road Vehicle Automation 5*, pages 131–147. Springer, 2019.

[4] Pedro Augusto, Jaime S. Cardoso, and Joaquim Fonseca. Automotive interior sensing - towards a synergetic approach between anomaly detection and action recognition strategies. In *Proceedings of Fourth IEEE International Conference on Image Processing, Applications and Systems (IPAS)*, 2020.

[5] Laura Sevilla-Lara, Yiyi Liao, Fatma Güney, Varun Jampani, Andreas Geiger, and Michael J Black. On the integration of optical flow and action recognition. In *German Conference on Pattern Recognition*, pages 281–297. Springer, 2018.

[6] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. Vancouver, British Columbia, 1981.

[7] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Scandinavian conference on Image analysis*, pages 363–370. Springer, 2003.

[8] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *Proceedings of the IEEE international conference on computer vision*, pages 3551–3558, 2013.

[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[10] Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 17–36, 2012.

[11] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3rd edition, 2009.

[12] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[13] Convolutional neural networks. https://www.mathworks.com/discovery/convolutional-neural-network-matlab.html.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[15] Alberto Fernández, Salvador García, Mikel Galar, Ronaldo C Prati, Bartosz Krawczyk, and Francisco Herrera. *Learning from imbalanced data sets*. Springer, 2018.

[16] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[17] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *2011 International conference on computer vision*, pages 2556–2563. IEEE, 2011.

[18] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[19] Mathew Monfort, Alex Andonian, Bolei Zhou, Kandan Ramakrishnan, Sarah Adel Bargal, Tom Yan, Lisa Brown, Quanfu Fan, Dan Gutfreund, Carl Vondrick, et al. Moments in time dataset: one million videos for event understanding. *IEEE transactions on pattern analysis and machine intelligence*, 42(2):502–508, 2019.

[20] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[21] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *arXiv preprint arXiv:1406.2199*, 2014.

[22] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

[23] Zhiwei Deng, Arash Vahdat, Hexiang Hu, and Greg Mori. Structure inference machines: Recurrent neural networks for analyzing relations in group activity recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4772–4781, 2016.

[24] Timur Bagautdinov, Alexandre Alahi, François Fleuret, Pascal Fua, and Silvio Savarese. Social scene understanding: End-to-end multi-person action localization and collective activity recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4315–4324, 2017.

[25] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1933–1941, 2016.

[26] Eunbyung Park, Xufeng Han, Tamara L Berg, and Alexander C Berg. Combining multiple sources of knowledge in deep cnns for action recognition. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–8. IEEE, 2016.

[27] Muhammad Usman Khalid and Jie Yu. Multi-modal three-stream network for action recognition. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 3210–3215. IEEE, 2018.

[28] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European conference on computer vision*, pages 20–36. Springer, 2016.

[29] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2012.

[30] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[31] Lin Sun, Kui Jia, Dit-Yan Yeung, and Bertram E Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4597–4605, 2015.

[32] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning for video understanding. *arXiv preprint arXiv:1712.04851*, 1(2):5, 2017.

[33] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *proceedings of the IEEE International Conference on Computer Vision*, pages 5533–5541, 2017.

[34] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018.

[35] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

[36] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[37] Yang Xing, Chen Lv, Huaji Wang, Dongpu Cao, Efstathios Velenis, and Fei-Yue Wang. Driver activity recognition for intelligent vehicles: A deep learning approach. *IEEE Transactions on Vehicular Technology*, 68(6):5379–5390, 2019.

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[39] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[40] Martin Torstensson, Thanh Hai Bui, David Lindström, Cristofer Englund, and Boris Duran. In-vehicle driver and passenger activity recognition. In *37th Annual Swedish Symposium on Image Analysis (SSBA 2019), Gothenburg, Sweden, March 19-20, 2019*, 2019.

[41] Yong Shean Chong and Yong Haur Tay. Abnormal event detection in videos using spatiotemporal autoencoder. In *International symposium on neural networks*, pages 189–196. Springer, 2017.

[42] Alexandros Stergiou and Ronald Poppe. Analyzing human–human interactions: A survey. *Computer Vision and Image Understanding*, 188:102799, 2019.

[43] Pedro M Ferreira, Filipe Marques, Jaime S Cardoso, and Ana Rebelo. Physiological inspired deep neural networks for emotion recognition. *IEEE Access*, 6:53930–53943, 2018.

[44] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 961–970, 2015.

[45] Dhwani Mehta, Mohammad Faridul Haque Siddiqui, and Ahmad Y Javaid. Facial emotion recognition: A survey and real-world user experiences in mixed reality. *Sensors*, 18(2):416, 2018.

[46] Lianzhi Tan, Kaipeng Zhang, Kai Wang, Xiaoxing Zeng, Xiaojiang Peng, and Yu Qiao. Group emotion recognition with individual facial emotion cnns and global image based cnns. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, pages 549–552, 2017.

[47] Abhinav Dhall, Garima Sharma, Roland Goecke, and Tom Gedeon. Emotiw 2020: Driver gaze, group emotion, student engagement and physiological signal based challenges. In *Proceedings of the 2020 International Conference on Multimodal Interaction*, pages 784–789, 2020.

[48] Bo Sun, Qinglan Wei, Liandong Li, Qihua Xu, Jun He, and Lejun Yu. Lstm for dynamic emotion and group emotion recognition in the wild. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction*, pages 451–457, 2016.

[49] Qinglan Wei, Yijia Zhao, Qihua Xu, Liandong Li, Jun He, Lejun Yu, and Bo Sun. A new deep-learning framework for group emotion recognition. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, pages 587–592, 2017.

[50] Aarush Gupta, Dakshit Agrawal, Hardik Chauhan, Jose Dolz, and Marco Pedersoli. An attention model for group-level emotion recognition. In *Proceedings of the 20th ACM International Conference on Multimodal Interaction*, pages 611–615, 2018.

[51] Evangelos Kazakos, Arsha Nagrani, Andrew Zisserman, and Dima Damen. Epic-fusion: Audio-visual temporal binding for egocentric action recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5492–5501, 2019.

[52] Robin Cosbey, Allison Wusterbarth, and Brian Hutchinson. Deep learning for classroom activity detection from audio. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3727–3731. IEEE, 2019.

[53] Dawei Liang and Edison Thomaz. Audio-based activities of daily living (adl) recognition with large-scale acoustic embeddings from online videos. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 3(1):1–18, 2019.

[54] Wei Wang, Fatjon Seraj, Nirvana Meratnia, and Paul JM Havinga. Privacy-aware environmental sound classification for indoor human activity recognition. In *Proceedings of the 12th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, pages 36–44, 2019.

[55] Joao Ribeiro Pinto, Tiago Goncalves, Carolina Pinto, Luis Sanhudo, Joaquim Fonseca, Filipe Goncalves, Pedro Carvalho, and Jaime S. Cardoso. Audiovisual classification of group emotion valence using activity recognition networks. In *Proceedings of Fourth IEEE International Conference on Image Processing, Applications and Systems (IPAS)*, 2020.

[56] Garima Sharma, Shreya Ghosh, and Abhinav Dhall. Automatic group level affect and cohesion prediction in videos. In *2019 8th International Conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW)*, pages 161–167. IEEE, 2019.

[57] Florian Eyben. *Real-time speech and music classification by large audio feature space extraction*. Springer, 2015.

[58] Seyedmahdad Mirsamadi, Emad Barsoum, and Cha Zhang. Automatic speech emotion recognition using recurrent neural networks with local attention. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2227–2231. IEEE, 2017.

[59] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678, 2014.

[60] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[61] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[62] Vgg16 – convolutional network for classification and detection. https://neurohive.io/en/popular-networks/vgg16/.

[63] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[64] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[65] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[66] Timothy Dozat. Incorporating nesterov momentum into adam. 2016.

[67] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[68] Christopher Zach, Thomas Pock, and Horst Bischof. A duality based approach for realtime tv-l 1 optical flow. In *Joint pattern recognition symposium*, pages 214–223. Springer, 2007.