INSTITUTO
UNIVERSITÁRIO
DE LISBOA

Assessing the Emotional Impact of Video using Machine Learning Techniques

André Filipe Lopes Maia

Master in Telecommunications and Computer Engineering

Supervisor:

Doctor Tomás Gomes Silva Serpa Brandão, Assistant Professor,
Iscte – Instituto Universitário de Lisboa

Co-supervisor:

Doctor Fernando Manuel Marques Batista, Associate Professor,
Iscte – Instituto Universitário de Lisboa

December, 2020

TECNOLOGIAS
E ARQUITETURA

Assessing the Emotional Impact of Video using Machine Learning Techniques

André Filipe Lopes Maia

Master in Telecommunications and Computer Engineering

Supervisor:

Doctor Tomás Gomes Silva Serpa Brandão, Assistant Professor,
Iscte – Instituto Universitário de Lisboa

Co-supervisor:

Doctor Fernando Manuel Marques Batista, Associate Professor,
Iscte – Instituto Universitário de Lisboa

December, 2020

Resumo

Quando o ser humano assiste a filmes, diferentes sensações e estados de espírito são despoletados. Entre estes encontra-se o medo, que pode ser despoletado através da visualização de excertos de filmes contendo, por exemplo, violência gráfica, horror ou *suspense*. Tanto a componente visual como a auditiva contribuem para o despoletar desta sensação. Nesta dissertação é analisada a utilização de aprendizagem automática para prever o impacto emocional que a visualização de vídeos possa causar nas pessoas, mais concretamente os segmentos de um filme que despoletam a sensação de medo.

Foram realizadas diversas experiências usando o conjunto de dados *LIRIS-ACCEDE* com os objetivos de encontrar conjuntos de atributos de imagem e áudio com maior relevância para o problema e de avaliar o desempenho de diversos modelos de aprendizagem automática usados para classificação. Foram usados diversos algoritmos clássicos e de aprendizagem profunda, recorrendo-se às bibliotecas *Scikit-learn* e *TensorFlow*. No que se refere à separação dos dados usados para treino e teste foram seguidas duas abordagens: divisão dos dados ao nível do filme, sendo usados filmes distintos para treino e teste; e divisão dos dados ao nível da amostra, possibilitando que os conjuntos de treino e teste contenham amostras distintas, mas pertencentes aos mesmos filmes. Para previsão dos segmentos que despoletam medo, na primeira abordagem chegou-se a um resultado de F1-score de 18,5%, concluindo-se que o conjunto de dados usado não é representativo, e na segunda abordagem a um F1-score de 84,0%, um valor substancialmente mais alto e promissor no desempenho da tarefa proposta.

Palavras chave

Aprendizagem Automática, Predição Emocional, Predição de Medo, Classificação de Video

Abstract

Typically, when a human being watches a video, different sensations and mind states can be stimulated. Among these, the sensation of fear can be triggered by watching segments of movies containing themes such as violence, horror and suspense. Both the audio and visual stimuli may contribute to induce fear onto the viewer. This dissertation studies the use of machine learning for forecasting the emotional effects triggered by video, more precisely, the automatic identification of fear inducing video segments.

Using the *LIRIS-ACCEDE* dataset, several experiments have been performed in order to identify feature sets that are most relevant to the problem and to assess the performance of different machine learning classifiers. Both classical and deep learning techniques have been implemented and evaluated, using the *Scikit-learn* and *TensorFlow* machine learning libraries. Two different approaches for training and testing have been followed: film-level dataset splitting, where different films were used for training and testing; and sample-level dataset splitting, which allowed that different samples coming from the same films were used for training and testing. The prediction of movie segments that trigger fear sensations achieved a F1-score of 18.5% in the first approach, a value suggesting that the dataset does not adequately represent the universe of movies. The second approach achieved a F1-score of about 84.0%, a substantially higher value that shows promising outcomes when performing the proposed task.

Keywords

Machine Learning, Emotional Prediction, Fear Prediction, Video Classification

Acknowledgments

Esta tese não teria sido possível sem a ajuda de todas as pessoas que me são importantes e que me apoiaram nesta fase da minha vida. Aos meus pais, irmã, namorada, familiares e amigos mais próximos por me terem ajudado e motivado a progredir e concluir esta dissertação, por perguntarem sempre como estava a correr e por se interessarem sobre o que era. A todas as pessoas que fizeram parte e me ajudaram, de qualquer maneira, nesta dissertação, um grande obrigado.

Quero também agradecer aos meus orientadores, Tomás Brandão e Fernando Batista, por me terem ajudado no decorrer desta dissertação. Pelas inúmeras reuniões feitas para que no fim o resultado fosse o melhor, pelas ideias e melhorias a fazer, um obrigado. Um obrigado também por estarem sempre dispostos a ajudar sempre que existia alguma dúvida, e sempre disponíveis quando era necessário. Foi uma grande sorte ter encontrado orientadores que se preocupam com o aluno e com o resultado final da dissertação.

No final, termino esta dissertação com a sensação de dever cumprido.

Lisboa, 1 de Novembro de 2020

André Maia

Contents

1 Introduction **1**

1.1 Context and Motivation . . . 1

1.2 Proposed Work . . . 3

1.3 Objectives and Research Questions . . . 3

1.4 Research Method . . . 4

1.5 Dissertation Structure . . . 4

2 Machine Learning Concepts **7**

2.1 Classical Approaches . . . 7

2.1.1 Principal Component Analysis . . . 7

2.1.2 Neural Networks . . . 8

2.1.3 Support Vector Machine . . . 10

2.1.4 K-Nearest Neighbors . . . 10

2.1.5 Linear Discriminant Analysis . . . 11

2.1.6 Decision Tree Classifier . . . 12

2.1.7 Gaussian Naive Bayes . . . 12

2.2 Deep Learning . . . 12

2.2.1 Convolutional Neural Networks . . . 12

2.2.2 Recurrent Neural Networks . . . 13

2.3 Optimization Algorithms . . . 15

2.4 Pre-trained Models . . . 16

2.5 Tools and Libraries . . . 19

2.6 Assessment Metrics . . . 19

2.6.1 Mean Square Error . . . 19

2.6.2 Pearson Correlation Coefficient . . . 20
2.6.3 Intersection over Union . . . 20
2.6.4 Accuracy, Precision and Recall . . . 20
2.6.5 F1-score . . . 22
2.6.6 Receiver Operating Characteristics and Area Under the Curve . . . 22

3 Literature Revision and Datasets 25

3.1 Related work . . . 26
3.1.1 Framewise classification models . . . 27
3.1.2 Time-sequential models . . . 32
3.2 Summary . . . 34

4 Classical Classifiers and Feature Analysis 35

4.1 Experimental Setup . . . 35
4.2 Using Different Films for Training and Testing . . . 37
4.2.1 Individual feature sets . . . 37
4.2.2 Best feature sets combined . . . 39
4.2.3 All visual feature sets combined . . . 40
4.2.4 Class balancing with all visual features sets . . . 40
4.2.5 All features sets combined . . . 41
4.2.6 Summary . . . 42
4.3 Using Samples From Same Films for Training and Testing . . . 42
4.3.1 Individual feature sets . . . 43
4.3.2 Best feature sets combined . . . 44
4.3.3 All visual feature sets combined . . . 44
4.3.4 Class balancing with visual features sets . . . 45
4.3.5 All features sets combined . . . 46
4.3.6 Summary . . . 46
4.4 Fear-inducing Classifications . . . 47
4.5 Summary . . . 50

5 Deep Learning Classification 51

5.1 Baseline Neural Network 51
5.2 Deep Neural Networks 57
5.3 Recurrent Neural Network 61
5.4 Summary 62

6 Conclusions and Future Work **63**

Bibliography **65**

List of Figures

1.1 Valence vs arousal. 2

2.1 Visualizing data in different dimensions. 8
2.2 Neural network with two hidden layers. 9
2.3 Neural network with and without a dropout. 9
2.4 Possible hyperplanes vs. optimal hyperplane [22]. 10
2.5 KNN with K=3 [57]. 11
2.6 PCA vs LDA [54]. 11
2.7 Typical CNN architecture [58]. 13
2.8 Convolutional Layer. 13
2.9 Recurrent Neural Network [42]. 14
2.10 LSTM gates [42]. 14
2.11 Gradient Descent formula example [60]. 15
2.12 RMSProp example. 16
2.13 Residual Network. 17
2.14 VGG16 Architecture [46]. 18
2.15 Inception module [66]. 18
2.16 Mean Square Error.[69] 19
2.17 Heteroscedasticity vs homoscedasticity [36]. 20
2.18 Object detection using IoU [55]. 21
2.19 Different values of AUC. [44]. 23

3.1 Proposed framework [73]. 27
3.2 Fear recognition block diagram [6]. 29

4.1 Three frames from the movie 0. 47

4.2 Second segment of the movie 0. 47
4.3 Segment of the movie 16. 48
4.4 Gaussian predictions for visual features. 48
4.5 Audio features predictions. 49
4.6 Predictions using the best method. 49

5.1 Model created using TensorFlow. 51
5.2 Number of epochs by loss. 52
5.3 Variation of the loss, accuracy, precision and recall. 53
5.4 ROC curve. 54
5.5 Number of epochs by the loss, accuracy, precision and recall with class weights. 55
5.6 ROC with all the models. 56
5.7 Model with 10 Dense layers and 9 Dropouts. 58
5.8 Number of epochs by the loss, accuracy, precision and recall. 59
5.9 ROC of the model with 10 Dense layers and 9 Dropouts. 60
5.10 RNN model. 61

List of Tables

2.1 Example of an confusion matrix. . . . 21

3.1 Features used in five runs [73]. . . . 28
3.2 Valence-arousal results [73]. . . . 28
3.3 Fear results [73]. . . . 29
3.4 Valence and arousal, and fear subtasks results [6]. . . . 30
3.5 Results for Valence, Arousal and fear [30]. . . . 31
3.6 Softsampling with different features [76]. . . . 32
3.7 Valence and arousal results [35]. . . . 33
3.8 Runs and fear results described by the team [32]. . . . 34

4.1 *F*1-score (*no-fear/ fear*) achieved for each feature set. . . . 38
4.2 *Sc* features set results. . . . 38
4.3 GaussianNB confusion matrix of *sc* feature set. . . . 39
4.4 Best feature sets combined. . . . 39
4.5 GaussianNB confusion matrix of best feature set combined. . . . 39
4.6 All visual features combined. . . . 40
4.7 All visual features combined and classes were balanced. . . . 41
4.8 Visual and audio features combined. . . . 41
4.9 Best results for Gaussian Naive Bayes Classifier. . . . 42
4.10 Individual features (F1-score). . . . 43
4.11 KNN confusion matrix of *acc* feature set. . . . 43
4.12 Best features combined. . . . 44
4.13 KNN confusion matrix of best feature sets combined. . . . 44
4.14 All visual features combined. . . . 45

4.15 Results of balanced classes. 45
4.16 MLP confusion matrix of balanced classes. 45
4.17 All features sets combined results. 46
4.18 MLP confusion matrix all features sets combined. 46
4.19 Best results. 46

5.1 Confusion matrix. 54
5.2 Model with class weights. 56
5.3 F1-score using the model with 5 hidden layers. 57
5.4 Confusion matrix. 58
5.5 F1-score using the model with 10 Dense layers and 9 Dropouts. 60

Acronyms

ACC Auto Color Correlation

CEDD Color and Edge Directivity Descriptor

CL Color Layout

EH Edge Histogram

FC6 VGG16 fc6 layer output

FCTH Fuzzy Color and Texture Histogram

JCD Joint descriptor joining CEDD and FCTH in on Histogram

LBP Local Binary Patterns

SC Scalable Color

LDA Linear Discriminant Analysis.

KNN K-Nearest Neighbor.

MLP Multi-layer Perceptron

GaussianNB Gaussian Naive Bayes

LR Logistic Regression

SVM Linear Support Vector

AUC Area Under the Curve

IoU Intersection over Union

CNN Convolutional Neural Network

RNN Recurrent Neural Network

ML Machine learning

Chapter 1

Introduction

The purpose of this chapter is to provide some insight into the motivation and main goals of this dissertation, as well as the context, research questions and method.

1.1 Context and Motivation

Machine learning (ML) is a trend in the current days with new methodologies developed every moment. As its name suggests, the emphasis of ML is on learning, acquiring skills or knowledge from experience. It is important to have a large amount data to work with, since most ML approaches need data to learn. This type of technique is used to handle certain tasks that are incredibly difficult to create a program that has answers for everything, such as detecting the emotional impact of movies, spam filtering, face recognition, data mining, robot motion, among others, and can solve very complicated tasks taking much less time than a human [31].

When watching movies, feelings and states of mind are induced into the viewers. Since no one is identical, the induced emotions may differ from person to person. Human beings, according to scientists, can manifest up to twenty seven distinct emotions [59], and human emotions can be mapped into six different categories: Happy; Sad; Angry; Fear/Worry; Surprise; and Disgust, proposed by Ekman [14]. When watching a movie, these emotions can be invoked and can be manipulated with the assistance of combining sound and images. One example of manipulation is advertising, that attempts to transmit specific sensations to audiences to accomplish their objectives of selling the product itself. A variety of sensations are triggered during the course of a film, but fear is the emotion that is wanted and will be predicted using ML.

Valence and arousal are two measurements that can be used for characterizing emotions. Valence expresses how pleasant is the feeling while arousal expresses the level of excitement. Both can assume positive or negative values – unpleasant feelings correspond to negative valence values, while calm states correspond to negative arousal

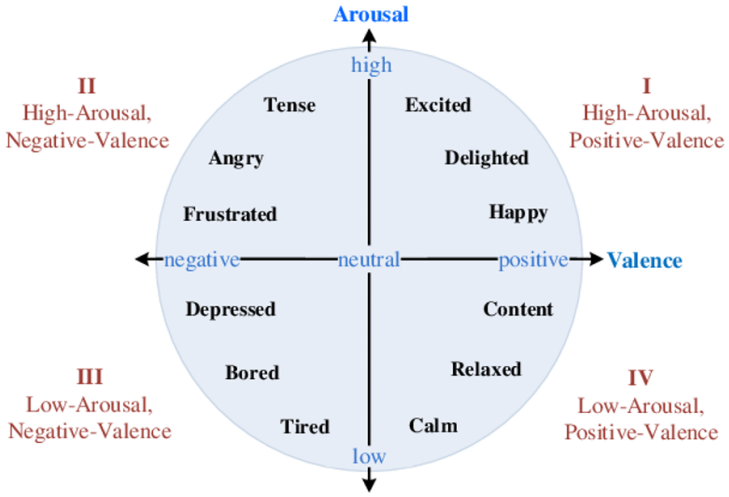

Figure 1.1: Valence vs arousal.

values. Figure 1.1, extracted from [75], represents an association between possible affective states and valence and arousal values.

The six categories of emotions mentioned previously can be mapped using these two measurements, for example, happiness can be expressed as a high value of valence and positive value in arousal.

Machine learning methods may potentially be used for automatically predicting the emotional effect of movies. This application can be very beneficial and has many advantages, such as:

- Video summarization mechanisms that take emotional factor into consideration, summarizing the video according to excerpts that represent emotional states.
- Improving the recommendation mechanisms and personalization of video transmission services. For instance, to evaluate a film in order to produce tags according to the triggered emotional states.
- Protect viewers from content that could affect them, such as children and more vulnerable people. For example, filter material to protect a child who surfs the Internet watching a variety of videos, by blocking content that may include violent or disturbing scenes.

Analyzing the emotional impact of a video clip on audiences may also be used to strengthen or monitor the psychological influence of media on persons, to maximize audience interaction with media material, or to create customized media content [52].

1.2 Proposed Work

The key purpose of this thesis is to predict the emotional effect of audiovisual content on individuals, and this may include a variety of similar applications, such as personalized content discovery or video indexing, film searching and even protecting children from videos that may potentially be dangerous [32, 6]. A dataset named *LIRIS-ACCEDE* [34], which provides a total of 56 movies, corresponding to approximately to 24 hours, with annotations focused on fear, valence and arousal, was used. This database already includes several sets of features, extracted for every second of each movie, such as color and texture visual features, audio features, etc., providing a total of twelve feature sets containing a large amount of features.

Different approaches are proposed in order to evaluate the results, which feature sets and ML models lead to the best results in predicting the fear inducing segments of the movies. Two separate training and testing methods are used: the first one is to split the data at film level, getting samples coming from different films for training and testing, while the second is less restrictive, randomly splitting the data at the sample level, causing samples of all movies to appear in the train and test sets.

Two Python libraries, called *Scikit-learn* and *TensorFlow*, are used for the estimation of fear inducing film segments. *Scikit-learn* classificiers are used for the implementation of classical ML models, such as Gaussian Naive Bayes, Decision Tree Classifier, Logistic Regression, among others; *TensorFlow* is used for the development of deep neural networks.

1.3 Objectives and Research Questions

The main goal of this dissertation is to automatically detect movie segments that may trigger the induction of fear, based on the characteristics of the audiovisual content. In order to perform this task, it is important to understand which ML algorithms perform the best predictions, namely to discover whether classical and deep learning ML algorithms (such as Convolutional Neural Networks and Recurrent Neural Networks) can be used to recognize fear inducing movie segments.

The work developed in the scope of this dissertation should allow to answer the following research questions:

1. What are the most relevant audio and feature sets for the fear detection task? Which feature sets can achieve the best outcome?

2. Which ML algorithms and techniques are best suited to predict fear-inducing movies scenes?

3. Do deep learning techniques lead to better outcomes, when compared with classical ML approaches?

1.4 Research Method

The research method followed in this dissertation is the problem-solving process Design Science Research methodology, which is composed of six activities [49]:

1. Identification of the problem and motivation – in this dissertation, the problem is to automatically detect the segments of a movie that may induce fear into the viewers mind. This task is performed with the help of machine learning.
2. Define objectives for a solution – the goal is to determine which algorithms and feature sets lead to the best fear prediction results.
3. Design and development – several machine learning models are created using the machine learning libraries *Scikit-learn* and *TensorFlow*. These models follow both classical and deep learning approaches.
4. Demonstration – the developed models have been used for detecting fear inducing movie segments, and their performance was compared between each other.
5. Evaluation – the performance of the developed models was evaluated using metrics typically employed in machine learning problems, namely the F1-score and the Intersection of Union (IoU). Additionally, the deep learning models are subjectively assessed with their learning and receiver operating characteristics (ROC) curves.
6. Communicate the problem and its relevance – this activity is accomplished by writing this dissertation, by reporting the results and the problems that have been encountered.

1.5 Dissertation Structure

This dissertation is structured as follows and is intended to represent the various phases of the work:

In chapter 2, Machine Learning concepts used during the dissertation are described, such as Neural Networks, pre-trained models, tools and libraries, and utilized metrics, among others.

Chapter 3 describes the literature review, presenting related work on methods and algorithms used to forecast aspects of the movie that induce fear in spectators. It also describes the datasets required for the execution and conclusion of the work.

Chapter 4 describes the first experiments performed, using a library in Python called *Scikit-learn*. This library contains the implementation of several machine learning algorithms that were tested in order to determine which lead to the best results. Along with the algorithm testing, the experimental work reported in this chapter also seeks to determine which feature sets are the most relevant for the prediction of fear inducing segments.

Chapter 5 consists of experiments similar to those carried out in Chapter 4, except in this case, deep learning methods have been used. These were implemented using *TensorFlow*. The results are presented and compared with the ones reported in Chapter 4.

Lastly, Chapter 6 describes the main conclusions of the study carried out within the framework of this thesis.

Chapter 2

Machine Learning Concepts

Machine learning (ML) is an artificial intelligence field that enables computer systems to learn from data samples instead of explicitly programming them. In short, the learning methods try to use patterns in the provided data in order to perform predictions with minimal human intervention [65].

This chapter presents the main machine learning concepts used and mentioned in the context of this dissertation. It starts by describing classical ML approaches such as Principal Component Analysis (PCA), Neural Networks (NN) and K-Nearest Neighbours (KNN). It also describes deep learning methods such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN), the main pre-trained CNN models and their use for transfer learning, and the tools and libraries necessary to create machine learning classifiers for classical deep learning approaches. The chapter ends with a description of the typical metrics used for assessing the performance of ML-based classification systems.

2.1 Classical Approaches

Although different and numerous machine learning techniques and methodologies exist, this section reviews the classical ML approaches that are relevant to the scope of this dissertation.

2.1.1 Principal Component Analysis

Principal Component Analysis is a technique that can be used for reducing the dimension of a feature space [21]. It identifies data patterns and expresses the data in such a way that the most important features can be highlighted. The main advantage of this technique is therefore the capacity of reducing the number of dimensions, transforming a large set of variables (features) into a smaller one, with minimal loss of information [63].

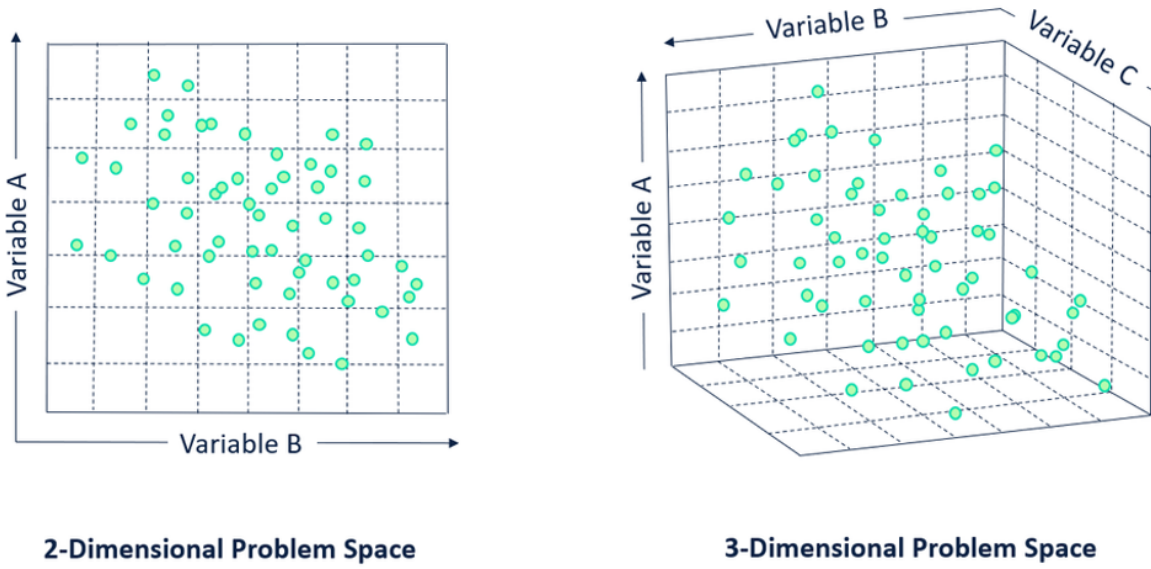

Figure 2.1: Visualizing data in different dimensions.

Imagine a problem with more than 3 dimensions (*i.e.*, more than 3 variables), for instance a 6-dimensional problem. Visualizing the data in a 6-dimensional space is very difficult, or even impossible. PCA may play a important role here, allowing the reduction of the 6-dimensional space into a 2-dimensional one, for instance, by considering less variables that carry almost the same information as the full variable set. That would enable easier data visualization and would eventually simplify the problem's solving. PCA can consider fewer variables by investigating a set of variables that are correlated to each other and, for example, merging all of them and consider only one variable.

Figure 2.1, extracted from [21], shows a 3 dimensional space with a set of variables. Using PCA, it is possible to reduce the dimensionality into a 2 dimensional set and at the same time keeping the most relevant information.

2.1.2 Neural Networks

A Neural Network is a learning technique that learns from sampled data. Data samples need to be previously labeled because training the NN requires knowledge about the class associated to each sample. For instance, if a random dog image is submitted to a NN for classification, the NN needs to have previous knowledge from other dog images in order to know what a dog image looks like. This type of learning methodology is called supervised learning (requiring labeled training and test sets). The creation of an artificial neural network via an algorithm allows the computer to learn by adding new data and is used for recognizing patterns [42, 17].

A Neural Network consists of several neuron units, organized into layers, that are used to perform predictions. A basic NN representation can be observed in Figure 2.2,

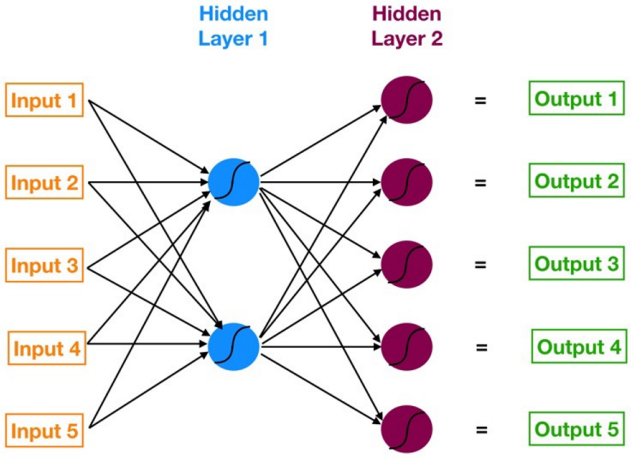

Figure 2.2: Neural network with two hidden layers.

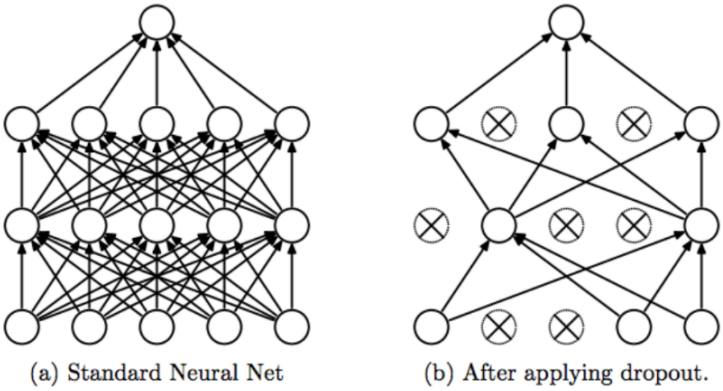

Figure 2.3: Neural network with and without a dropout.

extracted from [74]. It consists of an input layer (orange), that accepts the data for classification; two hidden layers (blue and magenta); and an output layer (green) that outputs the model's predictions. The output layer will have the same number of neurons as the output information (number of classes in the labeled data).

The arrows between each layer show how all the neurons are interconnected. The neuron's goal is to take all its inputs and multiply each one by their weights, then it sums them all and the resulting sum value is subject to an activation function [47].

The weights of a neuron implicitly decide which features are more important and, if a feature has higher importance, the result will have that feature more highlighted. Each connection has its own weight that starts with a constant or random value which is subsequently adjusted during the training of the model. Another important feature of the NN is the bias, which is a constant value associated with each layer that aims to shift the activation function.

Activation functions are used for standardizing the output values of each neuron. It is basically a function applied to the output. The most common activation functions are *Relu*

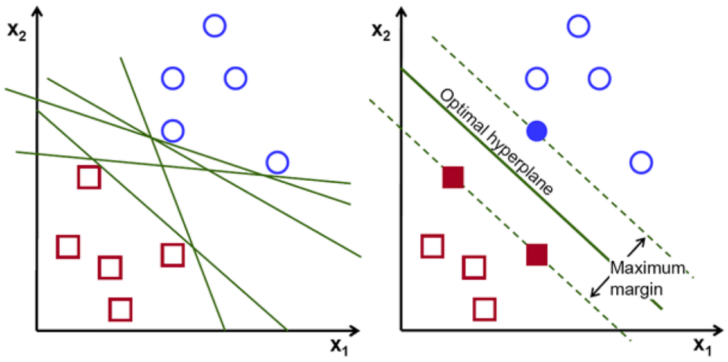

Figure 2.4: Possible hyperplanes vs. optimal hyperplane [22].

(Rectified Linear Unit), *Tanh* (Hyperbolic Tangent) and *Sigmoid* [56]. *Sigmoid* activation function transforms an input value into a an output value between 0 and 1, for instance, if the output value of a neuron is -8 the result will be very close to 0.

A Neural Network can have dropout between layers, which is simply used to ignore random neurons during the training phase. The neurons that will be ignored will not be considered during a forward or backward transition. At each point of training, the neurons are either dropped with a $1 - \rho$ probability or held with a ρ probability [9]. An example of the dropout is shown in Figure 2.3, extracted from [9].

2.1.3 Support Vector Machine

Support Vector Machine is a ML tool for classification and regression [39]. The objective of this algorithm is to find a hyperplane, in an N-dimensional space, that distinctly classifies the data points [22]. Figure 2.4 shows the behavior and the result of a SVM: the left plot represents different hyperplane possibilities for dividing the two data clusters; the right plot represents the optimal hyperplane that is found using SVM.

Support Vector Regression (SVR) is extension to SVM that is used for estimation, instead of classification. SVR can perform predictions that are far more complex than those provided by a linear regression model.

2.1.4 K-Nearest Neighbors

The K-Nearest Neighbors (KNN) is a clustering algorithm that groups samples with similar characteristics. It decides which samples have feature values that are close to each other [23] in order to be placed in a cluster where the samples share similarities between themselves (*i.e.*, belong to the same class). On the other hand, if two samples are not similar, their feature values are likely to be very different and therefore the clustering algorithm places them in different clusters.

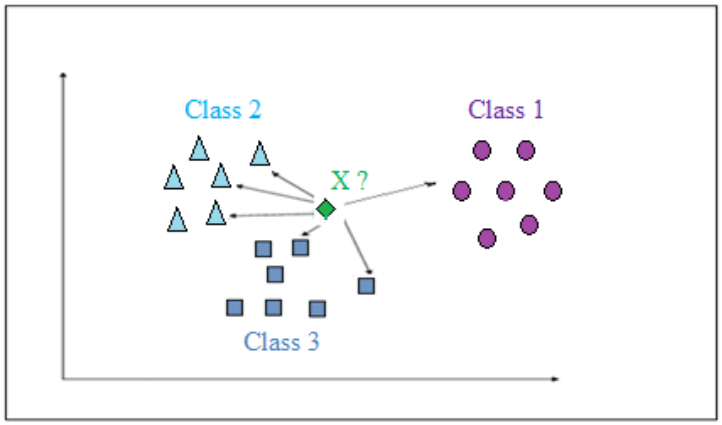

Figure 2.5: KNN with K=3 [57].

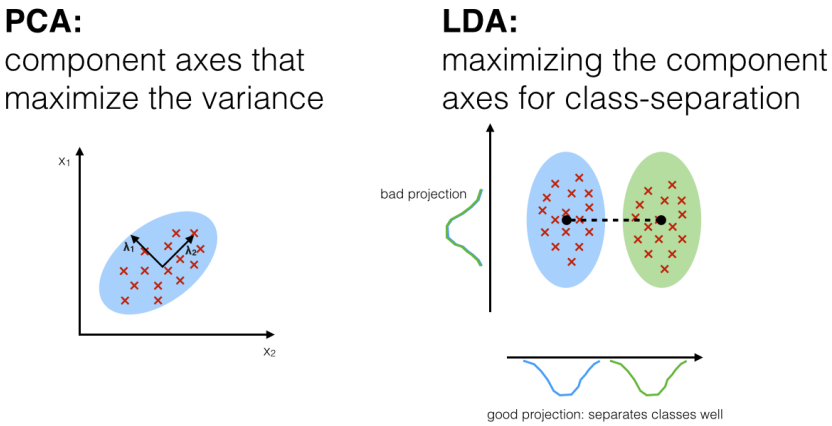

Figure 2.6: PCA vs LDA [54].

One of the problems with this method is to choose the value for K. The K tells how many clusters should be considered, which is basically the number of model classes. Figure 2.5 depicts an example where K=3, and thus the algorithm groups the data into three different classes.

2.1.5 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA), is closely related to PCA and is used for classification based on dimensionality reduction. The main objective is to separate classes in a dataset, lowering the dimensions and subsequently reducing the computational costs.

Figure 2.6 illustrates this separation, which is based on maximizing the component axes between two classes, instead of finding the axes that maximize the data variance, as PCA does [54].

2.1.6 Decision Tree Classifier

The Decision Tree Classifier is used for classifying examples based on decision rules. Decision Tree Classifier consists of a single root node, a different number of leaf nodes, and branches. Each node represents a feature and branches that reflect a combination of features contributing to that classification [20].

2.1.7 Gaussian Naive Bayes

Gaussian Naive Bayes can be used for binary classification (*i.e.*, with only two classes) as well as multi-class problems. It is based on *Bayes ' Theorem*, assuming that features are independent between each other, and that all features are equally weighted.

For two different events A and B, with $P(B) \neq 0$, *Bayes' Theorem* states that:

$$P(A|B) = \frac{P(A \bigcap B)}{P(B)} = \frac{P(A) \cdot P(B|A)}{P(B)}, \tag{2.1}$$

where $P(A)$ represents the probability of occurring event A , $P(A|B)$ is the probability of occurring event A given that event B occured, and $P(A \bigcap B)$ is the probability of both occurring.

Logistic Regression

Logistic Regression is a simple ML method that can be used for binary categorical classification [64]. Its output is a probability value (between 0 and 1) that can be used for predicting an object or event class. Instead of fitting a straight line as in linear regression model, the Logistic Regression uses a *Sigmoid* function to fit the data [77].

2.2 Deep Learning

Deep learning techniques are usually used on problems that are difficult to solve. This class of machine learning algorithms requires a large amount of labeled data and substantial computing power for the training procedures [41]. This section describes two classes of deep learning algorithms: Convolutional Neural Networks and Recurrent Neural Networks.

2.2.1 Convolutional Neural Networks

A Convolutional Neural Network is an artificial neural network whose architecture is designed to learn spatial hierarchies of features. A typical CNN architecture can be observed in Figure 2.7. It is mostly used in data that follows grid patterns, such as images. A CNN is generally composed of three different types of layers [72, 58]:

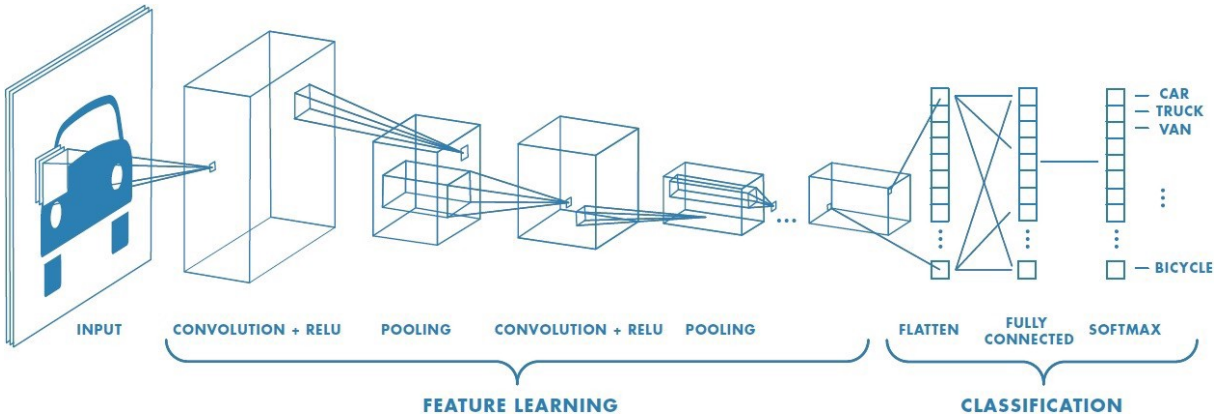

Figure 2.7: Typical CNN architecture [58].

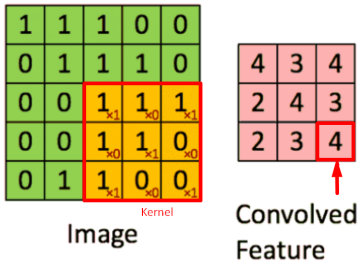

Figure 2.8: Convolutional Layer.

1. Convolutional layers – the purpose of these layers is to extract the high-level features, *e.g,*. edges, color, *etc* from the data matrices. A set of filter kernels (grids of parameters) are convoluted with the input, an image (a matrix of pixels), as shown in Figure 2.8 and activation functions are applied afterwards. The main goal is to find the filter kernels parameters, that highlight the most important high-level image features.

2. Pooling layers – these are responsible for reducing the spatial size of the matrices resulting from the convolutional layers. Different pooling possibilities can be used, such as *Max Pooling* and *Average Pooling*. The most typically used are *Max Pooling* layers, which return the maximum value found on non-overlapping windows of the input matrix (*e.g.*, the maximum value at each non-overlapping $x \times 2$ window).

3. Fully connected layers – a set of fully connected layers, whose structure is similar to a classical NN, receives the features resulting from the convolutional/pooling parts of the network, and produces the network predictions at its output.

2.2.2 Recurrent Neural Networks

Recurrent Neural Networks belong to a class of neural networks that have internal memory and are useful for modeling sequence data, such as time series [**TensorFlowCore2020**,

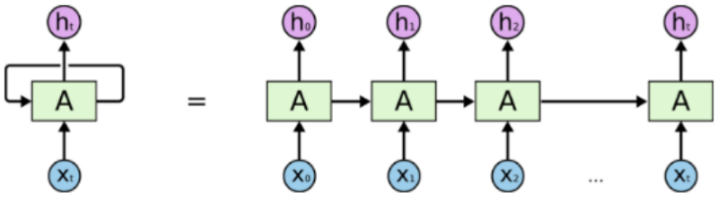

Figure 2.9: Recurrent Neural Network [42].

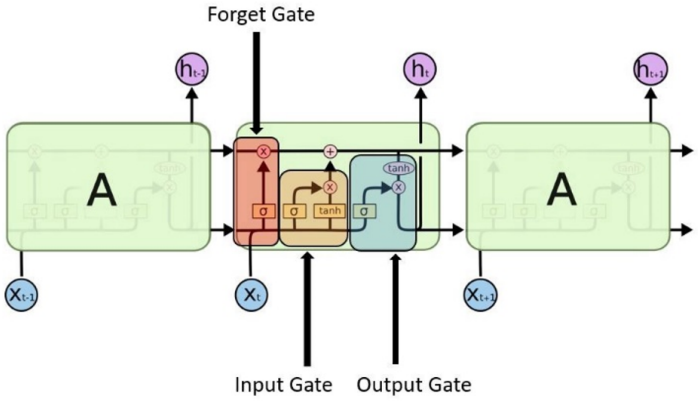

Figure 2.10: LSTM gates [42].

42]. RNNs are recurrent, which means that the output for the current input sample depends on the previous computations. The decisions are performed by considering the current input sample and the output that was learned from the previous input samples, as shown in Figure 2.9. Therefore, RNNs can use internal memory to process sequences of input samples. While other neural network types consider each sample independent of each other, Recurrent Neural Networks consider that the input samples are related to each other [42].

Long Short-Term Memory (LSTM) are a modified version of a basic Recurrent Neural Network that allow easier access to past data in memory. LSTM can classify, process and predict time series given time lags of unknown duration. The model is trained based on back-propagation [42].

Figure 2.10 represents the LSTM gates. The input gate decides which input value shall change the memory by using the *Sigmoid* and *Tanh* functions. The *Sigmoid* function is used to determine which values, between 0 and 1, are allowed to pass and the *Tanh* decide which values are more important by weighting them. The forget gate, with the *Sigmoid* function, determines which values should be discarded. The output gate selects which values are allowed to pass through, with the *Sigmoid* and *Tanh* functions. [42]

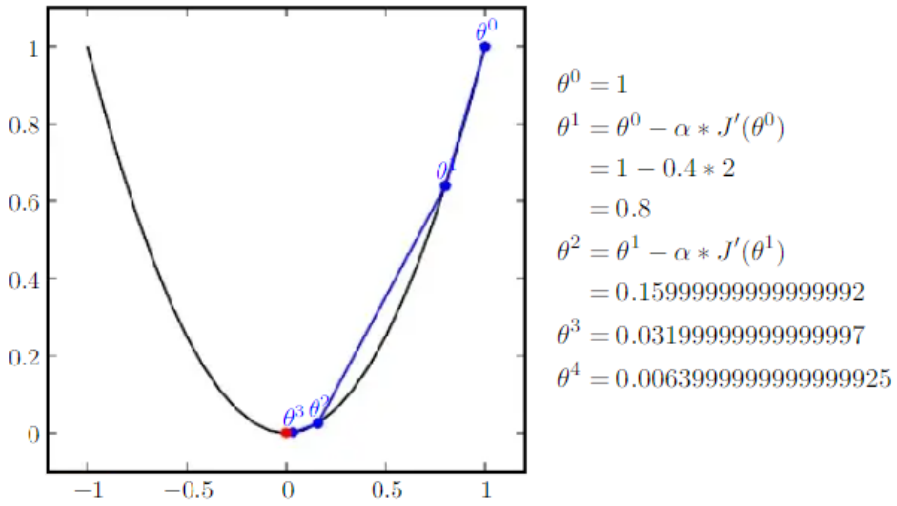

Figure 2.11: Gradient Descent formula example [60].

2.3 Optimization Algorithms

During the training process of a neural network, optimization algorithms are used in order to iteratively estimate the network parameters that lead to a minimal cost function value. In this section, a few optimization algorithms are described, namely *Gradient Descent, Root Mean Square propagation* (RMSProp), and *Adam*. *Gradient Descent* is the basic method for optimizing a neural network while *RMSProp* and *Adam* are variants that usually converge faster into the global function minimum.

Gradient Descent (GD) is one of the most popular algorithms for optimizing the training process of a neural network. This algorithm aims to find the optimal value for a given cost function.This optimal value corresponds to the minimum of the function, where its gradient is null. The iterative process followed by the algorithm can be summarized by the following equation:

$$\theta^{i+1} = \theta^i - \alpha \nabla J(\theta^i), \tag{2.2}$$

where θ^i and θ^{i+1} represent the current and next estimate of parameter θ and $\nabla J(\theta^i)$ represents the gradient of the function $J(\theta)$ under optimization, evaluated at the current parameter estimate; α is the learning rate or step size. Starting from an initial parameter guess θ^0 and iteratively applying eq. (2.2), GD will lead the estimates of θ to the value that minimizes function $J(\theta)$.

An example of using Gradient Descent is illustrated in Figure 2.11. The idea is to find the minimum of function $J(\theta) = \theta^2$ starting from $\theta^0 = 1$, with a step $\alpha = 0.4$. As can be be observed from the figure, after four iterations using eq. (2.2), the minimum value is almost reached.

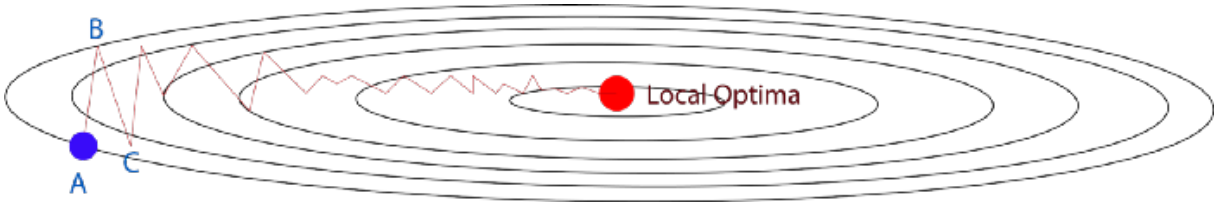

Figure 2.12: RMSProp example.

Root Mean Square propagation is another optimization method widely in machine learning, whose behavior is based on the Gradient Descent [15]. The differences from GD are the use of the gradient signal only, instead of its amplitude, and the use of an adaptive learning rate value. High fluctuations in the gradient's amplitude are therefore avoided and the use of an adaptive learning rate value ensures that it is adjusted at the current position by considering the shape of the function to be optimized.

The cost at the training start will be higher, but then RMSProp performs a faster approach to the global minima (where the cost function achieves the least possible value), calculating the negative descent towards the "Local optima", as shown in Figure 2.12, extracted from [61].

Adam attempts to automatically adjust the learning rate to various parameters, based on gradient statistics [24]. Adam combines the advantages of two other optimization algorithms: Root Mean Square Propagation (already mentioned); and the Adaptive Gradient Algorithm (AdaGrad) [7], which retains low learning rate values for features that appear regularly and higher learning rate values for features that appear less regularly [33].

2.4 Pre-trained Models

A pre-trained model is a neural network model containing the weights and biases adjusted to the features values of the dataset it was trained on [28]. This section describes a set of pre-trained CNN models, ResNet-50 and VGG16, which are typically used for transfer learning.

Transfer learning is a technique in machine learning that consists of using information learned to solve a given problem into another, different, yet related problem. The primary objective of transfer learning is therefore to profit from the learning acquired from one task into another related task [2]. Pre-trained models are used, since these models were typically trained on a wider dataset for solving a more generic problem [37].

Pre-trained models, such as ResNet-50 and VGG16, can be adapted to another problem, typically by substituting their fully connected network layers, while keeping the convolutional and pooling layers. Therefore, only the fully connected network layers need

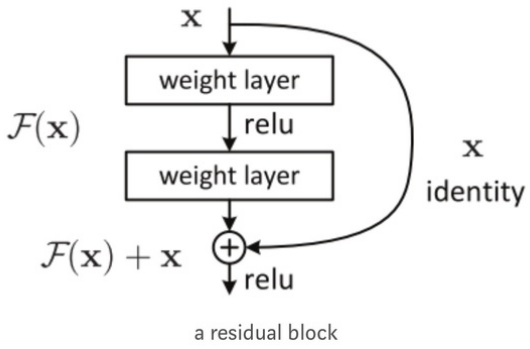

Figure 2.13: Residual Network.

to be trained in a transfer learning process, which may lead to important time savings during the network's training procedures [2].

ResNet means Residual Network and the number "50" refers to the number of layers on its architecture. It is a subclass of ResNet and is widely used for image classification [53].

The Residual Network concept consists of trying to learn residuals, instead of trying to learn features. The residual can simply be understood as a subtraction from the input of that layer of the feature learned. ResNet skips connections by directly connecting the layer a to layer $a+k$ (where a and $a+k$ are layers in ResNet, and k is the number of layers to skip) [53]. Figure 2.13, from [53], shows a skip connection, where the input can pass through the weight layers since the network learns the identity function (skip connection). The value x from the previous layer is added to the output of the layer ahead, $F(x)$, resulting in the output value of $F(x) + x$. This is very useful since a deeper network can be created, and this deeper network can skip over layers that the model finds less important.

According to [46], VGG16 is a convolutional neural network model that achieved 92.7% accuracy in the ImageNet challenge [18]. The network's input is a 224×224 RGB image. The image is then passed to a stack of convolutional layers, where the filters have a small 3×3 kernel. The network uses Max Pooling on its pooling layers and its ending is a set of fully connected layers where last layer is a soft-max one. The VGG16 architecture can be observed in Figure 2.14.

Inception network, described in [66], is a convolutional neural network model, which is composed by a concatenation of convolution filters with max-pooling layers. Each inception module receives the output of the previous layers in the input, and then a convolution is performed in each filter. The output results in a concatenation of the filter plus the max-pooling. Figure 2.15 represents an inception module containing 3 different filters and max pooling. One of the advantages of this architecture is to enable an increase of features without harming the computational complexity.

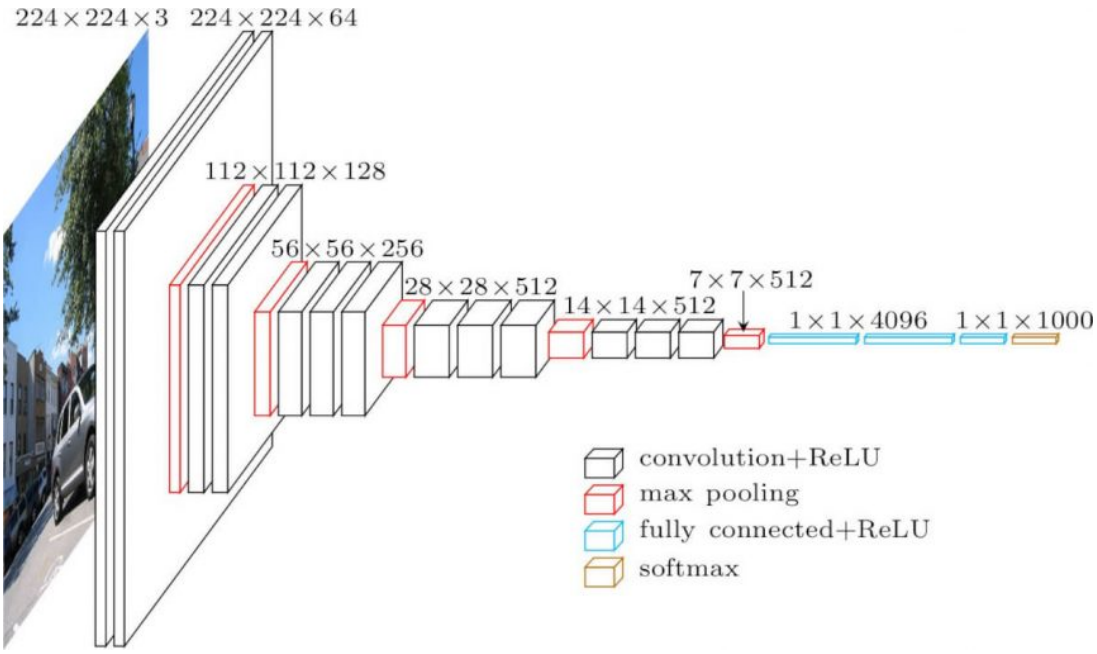

Figure 2.14: VGG16 Architecture [46].

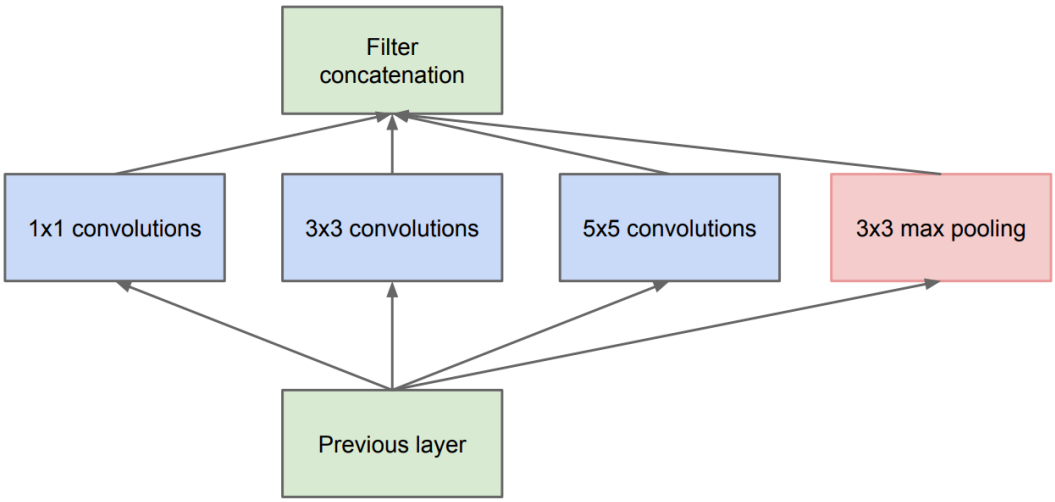

Figure 2.15: Inception module [66].

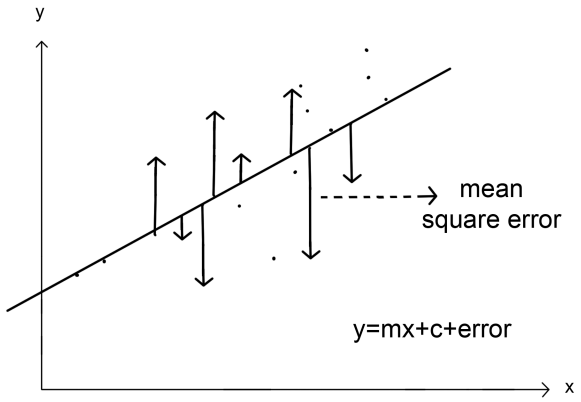

Figure 2.16: Mean Square Error.[69]

2.5 Tools and Libraries

The main tools and libraries for machine learning used for the work on this dissertation are briefly described in this section, namely *TensorFlow*, *openSmile* and a Python library called *Scikit-learn*.

TensorFlow [38] is a Google-created free software library focused on machine learning, deep learning and other statistical and predictive analytic workloads.

OpenSMILE is a tool that allows real-time extraction of a large number of audio features. It can be used for several applications such as automatic speech recognition, speaker identification, emotion recognition, or beat tracking and chord detection [45, 3].

Scikit-learn is a Python module with easy and powerful prediction implementation methods. It includes the implementation of several classical ML algorithms, including those described on section 2.1.

2.6 Assessment Metrics

Machine learning algorithms have different approaches for measuring their performance. In order to judge the performance of a model, an assessment metric function is used [27]. This section describes some of them.

2.6.1 Mean Square Error

Mean Square Error (MSE) identifies how close a regression line is to a set of points. This is achieved by taking the distances ('errors') from the points to the regression line and squaring them, eliminating some derogatory signs and also emphasizing larger errors [68].

When the MSE is small, a closer fit line is reached. Figure 2.16 represents the best line fitted in data.

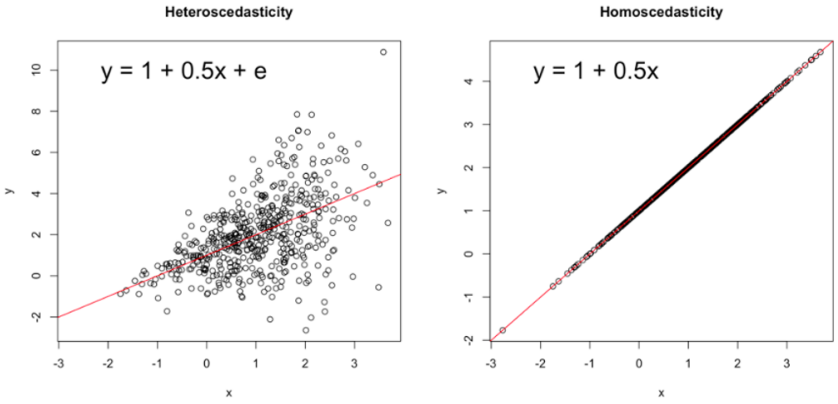

Figure 2.17: Heteroscedasticity vs homoscedasticity [36].

2.6.2 Pearson Correlation Coefficient

Correlation is the strength of association between two variables and the direction of the relationship. Its value ranges between -1 and 1. A value close to 0 indicates a weak relationship between variables while values that are close to 1 (or -1) means that the variables are strongly related. The minus sign shows a negative relationship and the plus sign the negative relationship [36].

Pearson correlation coefficient (PCC) measures the strength of a linear association between two variables [36]. Figure 2.17 illustrates a case where the variables are well correlated (homoscedasticity) and a case with low correlation (heteroscedasticity).

2.6.3 Intersection over Union

Intersection over Union (IoU), also known as Jaccard Index, is a metric of similarity that quantifies how similar two sets are to each other [10]. Equantion 2.3 depicts the Jaccard Index computation formula.

$$J(A, B) = \frac{|A \cap B|}{A \cup B} \tag{2.3}$$

In the context of image analysis, this metric is typically used for can measuring the performance of an object detector. An example can be observed in Figure 2.18 [55].

2.6.4 Accuracy, Precision and Recall

Accuracy, Precision and Recall are metrics commonly used in ML. A confusion matrix, represented in Table 2.1, can be used to compute these metrics. In a binary classification

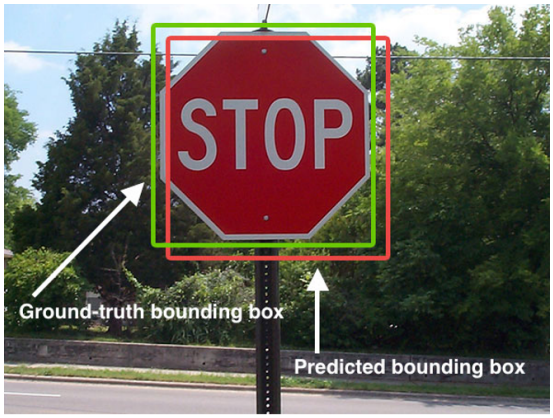

Figure 2.18: Object detection using IoU [55].

		Predicted	
		Positive	*Negative*
Actual	*Positive*	0 (TP)	9 (FP)
	Negative	0 (FN)	9999 (TN)

Table 2.1: Example of an confusion matrix.

problem, *Positive* represents one class while *Negative* represents the other. Samples that correctly classified are designated by True Positives (TP) or True Negatives (TN), depending to the class they belong to; samples that are misclassified are designated by False Positives (FP), if the actual class is *Negative* but the predicted class is *Positive;* and False Negatives (FN) otherwise.

Accuracy is the ratio between the number of correct predictions and the total number of performed predictions. In the case of Table 2.1, the accuracy is 99.9%, which appears to be a very good outcome. However, under this circumstance, this result is mischievous since, by evaluating the confusion matrix, it can be observed that the only class with correct predictions is the *Negative*. All predictions regarding the *Positive* class are incorrect. In addition, the confusion matrix shows that the data is unbalanced, since there are far more samples belonging to the *Negative* class than to the *Positive* class.

The Accuracy is computed as:

$$Accuracy = \frac{TP + FP}{TP + TN + FP + FN}. \tag{2.4}$$

Precision is the ratio between the correct predictions for a given class and its number of samples [62]. It is computed as:

$$Precision = \frac{TP}{TP + FP}. \tag{2.5}$$

For instance, using eq. (2.5) to compute the precision for the two classes in Table 2.1, it can concluded that the model has a precision of 100% in the *Negative* class and 0% in the *Positive*.

Recall simply calculates how many true positives the model captured by labeling them as positives [62]. It can be computed as:

$$Recall = \frac{TP}{TP + FN}. \tag{2.6}$$

Applying eq. (2.6) to determine the recall for the two classes in Table 2.1 results in 99.9% for the *Negative* class and 0% for the *Positive*.

2.6.5 F1-score

The metric F1-score combines the precision and the recall, according to:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.7}$$

F1-score is used when a trade-off between precision and the recall is required. When class distribution is unbalanced, the F1-score might be a more reliable metric to use.[62].

2.6.6 Receiver Operating Characteristics and Area Under the Curve

The Receiver Operating Characteristics (ROC) curve tells how well one model is capable of separating the classes. ROC is the probability curve, and the Area Under the Curve (AUC) refers to the area underneath the ROC curve [44, 19].

Ideally, an AUC equal to 1 means that the model performs a perfect distinction between the two classes [44]. This is the optimal case achieved where the ROC is constant and equal to 1, meaning that all samples of a given class can be correctly classified without generating False Positives. Figure 2.19 illustrates the AUC for different values. For instance, the model can distinguish classes if the AUC is between 0.5 and 1.0 because it is often possible to have a higher true positive rate than a false positive rate. The closer the AUC is to 1.0, the better this differentiation gets. A value of 0.5 means that the model can not differentiate the two classes, and a value of 0 means that the algorithm forecasts the negative class as the positive class, and vice versa.

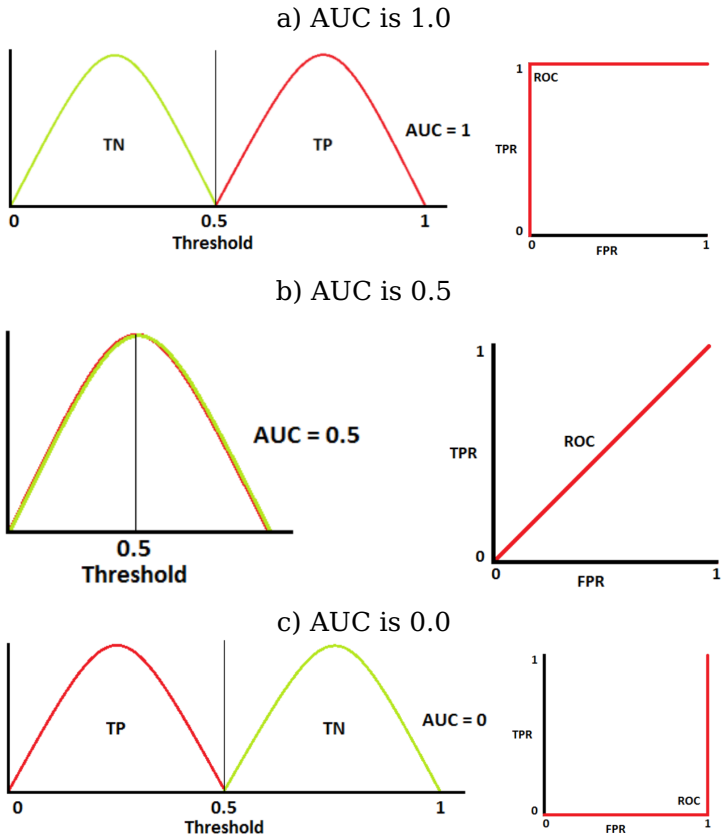

Figure 2.19: Different values of AUC. [44].

Chapter 3

Literature Revision and Datasets

This chapter starts by provide details for the datasets used in the mentioned works and then, it proceeds to briefly describing the related work, organizing it according to the followed methodologies.

The *LIRIS-ACCEDE* dataset is the main dataset used in the related work and in this dissertation. It consists of a collection of 160 films made by professionals and amateurs, shared under Creative Commons licenses that allow content redistribution. The movies in this dataset comprise a wide variety of genres such as suspense, satire, romance, action, *etc.* The language spoken in the videos is mainly English, with Italian, Spanish and French also spoken in a narrower video set. From the full film set, 56 which contains annotations regarding the fear-inducing movie parts (in total 4.2% of the samples contain *fear*), as well as values for valence and arousal are the ones used.

The dataset also provides general-purpose audio and visual content features. Audio features have been extracted with the openSmile toolbox, considering a 5-second window sliding through the entire movie, with one-second shifts, leading to one audio feature vector per video second, resulting in 1583 features. Video features, were computed from key frames extracted from every second of video [16]. The following sets of visual features have been provided:

1. Auto Color Correlation (*acc*): 256 features representing the image color correlogram [51].
2. Color and Edge Directivity Descriptor (*cedd*): 144 features that incorporate image color and texture information in a histogram [11].
3. Color Layout (*cl*): 33 features representing the spatial distribution of color in the image [71].
4. Edge Histogram (*eh*): 80 features representing information regarding five possible edge types in an image (horizontal, vertical, two diagonal and non-directional edge types) [48].

5. VGG16 fc6 layer output (*fc6*): 4096 features that correspond to the values extracted at the output of the first fully connected layer (fc6 layer) of the VGG16 convolutional neural network [40].
6. Fuzzy Color and Texture Histogram (*fcth*): 192 features representing the color similarity of each pixel color combined with the histogram bins [12].
7. Gabor (*gabor*): 60 image texture features [5].
8. Joint descriptor joining CEDD and FCTH in on Histogram (*jcd*): 168 features corresponding to 7 color texture areas of 24 sub-regions [13].
9. Local Binary Patterns (*lbp*): 256 features that represent image texture [70].
10. Scalable Color (*sc*): 64 features representing color signatures [26].
11. Tamura (*tamura*): 18 features representing statistical texture features [4].

ImageNet and *Places365* are other image datasets mentioned in the related work. *ImageNet* is often used as an image library containing tens of millions labeled images categorized into more than on hundred thousand classes [25]. *Places2* dataset has a total of about 10 million images classified into 400 different categories [50]. *Places365* is a subset of *Places2* dataset containing about 8 million images classified into 365 classes that represent scene types [29].

3.1 Related work

This section describes related work on assessing the emotional impact due to video visualization, namely evaluating whether the visualization of video induces fear on the viewer. Published works on this subject use several classical and deep learning-based machine learning models.

The starting point for this dissertation were the ideas behind a “benchmarking initiative dedicated to evaluating new algorithms for multimedia access and retrieval” called *MediaEval* [43]. *MediaEval* organizes several challenges on image and video recognition tasks. Among them, the one called *The MediaEval 2018 Emotional Impact of Movies Task* is very close to the purpose of this dissertation, and therefore it will be further analyzed and studied.

The main objective *MediaEval 2018 Emotional Impact of Movies Task* is to create a machine learning based program that automatically predicts the emotional impact that video content causes in its viewers [16]. The task’s participants are supposed to deploy multimedia features and models that allow to predict the emotion felt by most of the audience watching the movie. Each participant team should consider two scenarios as sub-tasks:

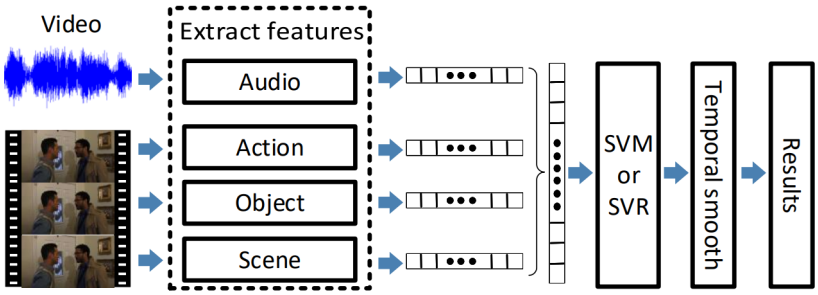

Figure 3.1: Proposed framework [73].

1. Valence and Arousal prediction: it's expected to continuously predict a score of valence and arousal (*i.e.*, every second) along the duration of the movie subject to analysis. Valence is classified as a scale, from the most unpleasant to the most pleasant emotional state, while Arousal is quantified from the calmest to the most exciting emotional states;

2. Fear detection: the participants are required to present a method that allows to predict the beginning and ending times of fear inducing sequences within the movies.

Different teams proposed different methods to handle the challenge. These approaches will be briefly described in the following subsections. Methods that classify each sample individually from the remaining ones are described in section 3.1.1, and approaches that take advantage of potential temporal dependencies of the samples, classifying sample sequences and not isolated ones, are detailed in section 3.1.2.

3.1.1 Framewise classification models

The approach proposed in [73] uses four sets of features that could influence emotions: audio, action, object and scene features, which are extracted using pre-trained convolutional neural networks. Figure 3.1 represents the proposed framework.

Audio signals are known to carry valuable knowledge for emotion identification [73]. The well-known audio feature extractor, VGGish [38], is used to extract the audio feature vector descriptors. After extracting the audio files from the video, the pre-trained model for large-scale audio classification is utilized to calculate their feature vectors. Then, VGGish converts the audio files to high-level 128-dimensional feature vectors. As for the action features extraction, a CNN is used, which contains two separate recognition streams, spatial and temporal. The result of the CNN are two 1024-dimensional feature vectors and, by merging these vectors, each video frame becomes associated to a 2048-dimensional feature vector. Object features are obtained by a CNN called the Squeeze-and-Excitation Network (SENet) and the result is a set of 2048 object features.

	Used features
Run 1	Features provided by *Mediaeval*
Run 2	Audio and scene features
Run 3	Audio, scene and object features
Run 4	Audio, scene and action features
Run 5	Audio, scene, object and action features

Table 3.1: Features used in five runs [73].

Run	Valence		Arousal	
	MSE	PCC	MSE	PCC
1	0.09142	0.27518	0.14634	0.11571
2	**0.09038**	**0.30084**	**0.13598**	0.15546
3	0.09163	0.26326	0.14056	0.14310
4	0.09105	0.25668	0.13624	**0.17486**
5	0.09243	0.24679	0.13950	0.15226

Table 3.2: Valence-arousal results [73].

To extract the scene features, the pre-trained ResNet-50 model on Places365 is used to calculate the vectors that result in a 2048-dimensional vector.

When all the features are extracted, the feature vectors are fused using SVR and SVM to learn the emotional models for each sub-task (fear, and valence and arousal) and Gaussian blur is used for smoothing the scores over the temporal segments. For each of the sub-tasks, different feature set combinations were tested, as described in the five runs depicted in Table 3.1.

The valence and arousal sub-task used SVR for regression and the metrics used for evaluation were the MSE and the PCC.

By observing Table 3.2, it is possible to infer that Run 2 produced the best results, which means that the combination of audio and scene features were the most adequate for valence and arousal estimation. The values that achieve the best results are in bold.

For the fear sub-task, a SVM was used for classification and the results were evaluated using the Intersection over Union metric. The results are shown in Table 3.3.

The best result was obtained by Run 4, using audio, scene and action features. It can also be observed that the combination of further features in Run 5 did not lead to better results.

Comparing the results of Run 2 and Run 3 in Tables 3.2 and 3.3, it can be inferred that the use of the object feature lowers the performance of the valence-arousal sub-task, but

Run	IoU
1	0.14360
2	0.12900
3	0.13067
4	**0.15750**
5	0.14969

Table 3.3: Fear results [73].

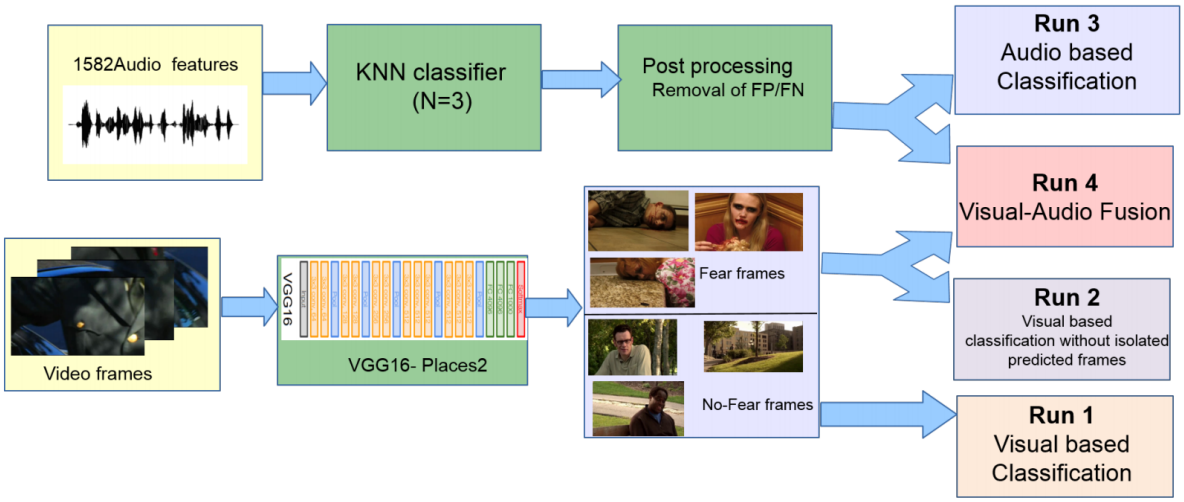

Figure 3.2: Fear recognition block diagram [6].

increases the performance for the fear sub-task. This may suggest that the presence of specific objects, such as a gun, a knife, *etc.*, may contribute to the induction of fear. Since Run 4 performed better than Run 3, actions may potentially have an higher impact on fear induction than objects.

A different approach was presented in [6]. This approach follows the block diagram represented in Figure 3.2.

Transfer learning using the pre-trained VGG16 model for Places2 dataset was used, since it would be expected that fear induction could be influenced by the movies background and scenery. The classification is performed for keyframes extracted on every second of video. Since the number of samples annotated as *fear* was much smaller than those annotated as *no-fear*, an higher amount of samples for the *fear* class were retrieved by downloading Flickr images tagged as *fear* and by using data augmentation techniques. It resulted in 23,000 and 30,000 images tagged as *fear* and *no-fear*, respectively, that were used for model training.

The *openSmile* toolbox was used to explore audio content by extracting 1582 audio features for each second of video. The samples were split by using 80 percent for training

Run	Valence		Arousal		Fear
	MSE	r	MSE	r	IoU
1	396901706.564	0.079	1678218552.19 0	**0.054**	**0.075**
2	0.139	0.010	0.181	-0.022	0.065
3	**0.117**	**0.098**	**0.138**	nan	0.053
4	0.142	0.067	0.187	-0.029	0.063

Table 3.4: Valence and arousal, and fear subtasks results [6].

and 20 percent for validation. The KNN classification method with K=3 and further processing were applied to the test set, in order to remove probable false negatives and false positives, *i.e.*, isolated predictions of *no-fear* inside *fear* intervals and isolated *fear* predictions inside *no-fear* intervals, respectively. The individual and combined use of visual and audio features were also evaluated.

The set of visual features provided by *LIRIS-ACCEDE* was also used. PCA was used to reduce the dimensionality of this feature space, downsampling the initial high dimensional feature space (5367 elements) into two thousand principal components.

The four different runs represented in Figure 3.2, were defined as follows: Run 1 uses the previously described transfer learning approach based on the VGG16 model; Run 2 is similar to Run 1, with additional post-processing to remove isolated predictions; Run 3 is based on audio features only, using a neural network with 3 hidden layers for classification; Run 4 also uses a NN combining the audio features with the visual features provided in *LIRIS-ACCEDE* (subject to PCA). The neural network used in Runs 3 and 4 uses ReLU activation function units and was trained with a learning rate of = 0.001 using the Adam optimizer. Each hidden layer of neurons has the size of 64, 32 and 32 respectively. Two thirds of the samples in the development set were used for training; the remaining one third of the samples were used for validation.

The results for the four runs are depicted in Table 3.4, with the best highlighted in bold. Mean Square Error (MSE) and Pearson Correlation Coefficient (r) were used for the assessment of valence and arousal prediction sub-task, while IoU was used in the fear sub-task. For the valence and arousal sub-task, Run 3 (based on a NN whose inputs are audio features only) was the one leading to the best results, when compared with the remaining methodologies. Run 1 obtained the best outcome for the fear detection sub-task, using only video based classification with the pre-trained VGG16 model.

In the work proposed by Ko *et al.* in [30], Emotion Preservation Embedding (EPE) was used to learn the subspace for arousal and valence, and a Bias Discriminatory Embedding algorithm (BDE) was used to learn fear.

Given a training set, EPE aims to learn a transformation matrix that maps each sample into a low-dimensional subspace, where the emotion information and manifold structure of

	Arousal		**Valence**		**Fear**
	MSE	PCC	MSE	PCC	IoU
Run 1	**0.1493**	**0.0828**	**0.1016**	0.0499	**0.1052**
Run 2	0.1574	0.0650	0.1089	0.0164	0.0612
Run 3	0.1608	0.0487	0.1089	0.0872	0.0360
Run 4	0.1623	0.0255	0.1076	**0.1142**	0.0196

Table 3.5: Results for Valence, Arousal and fear [30].

the dataset can be well preserved [30]. BDE attempts to learn the subspace for binary fear labels that increases the biased discriminatory information in the trained subspace.

A D-dimensional target subspace with $D = 4, 5, 9, 10$ was used for valence / arousal prediction in Runs 1, 2, 3 and 4, respectively, with a created D-dimensional subspace compacting the information equivalent to 2854 features. For fear prediction, the same values of valence / arousal prediction were used, except in this case BDE was used instead of EPE.

By analyzing Table 3.5, it can be concluded that Run 1 ($D = 4$) performed the best for arousal estimation. There is no optimum valence estimate for the valence sub-task, perhaps because the optimum valence subspace dimension has not been found [30]. Besides that, since MSE showed identical performance in all runs, while PCC achieved the highest result in Run 4 ($D = 10$), it can be concluded that this run performed the best. For the fear sub-task, the outcome was quite unsatisfactory, which may be due to a high dataset imbalance between the *fear* and *no-fear* classes. The best results are highlighted in bold

The approach presented in [76] consists of data re-sampling. This approach is performed for *MediaEval 2017 Emotional Impact of Movies Task*. Audio, visual and Valence-Arousal features were extracted resulting in 4172 features. To address the imbalanced dataset, different sampling methods were applied to the data:

1. All available data (without rebalance).
2. Applying a technique called Synthetic Minority Oversampling Technique (SMOTE), which will produce *fear* samples until the number of samples of *fear* became the same as *no-fear*.
3. Random Sampling.
4. Hardsampling, which applies undersampling before oversampling.
5. Softsampling, which applies first undersampling and then oversampling.

Then each sampling method was used to feed the two classifiers, Support Vector Machine (SVM) and Random Forest (RF). Lastly, late fusion is used. The best result using

Features	F1-score
Audio+Visual+Valence-Arousal	**0.3246**
Audio+Visual	0.3090
Valence-Arousal	0.2412
Audio	0.2353
Visual	**0.2907**

Table 3.6: Softsampling with different features [76].

all features (a combination of audio, visual, and Valence-Arousal.) was 0.3246 of F1-score, with the Softsampling. As the best result was obtained by Softsampling, this sampling method was used for the various features, as shown in Table 3.6.

Results shown, in Table 3.6, that visual features obtained the best results achieving 0.3246 of F1-score.

3.1.2 Time-sequential models

The approach proposed in [35] uses two features sets, one for audio and another for video. The audio features were extracted using the OpenSMILE toolbox, then the mean and standard deviation were calculated in the centered 5-second-long sliding window of all 23 features (low level descriptors) to achieve a 46-dimensional audio feature vector for each second of the movie clip. The audio features in the *LIRIS-ACCEDE* dataset (1582 dimensions) are also considered.

The visual feature sets provided by the *LIRIS-ACCEDE* dataset, except the *fc6* set, have also been considered, resulting in 1271-dimensional visual feature vector for each second of video. The *fc6* feature set was not used due to its high dimensionality (4096 features).

SentiBank was used for the extraction of additional visual features. The Multilingual Visual Sentiment Ontology (MVSO) detectors were applied to the image frames extracted every second from the film, and obtained the final layer of the Inception Net (CNN classifier), resulting in 4342 dimensions. All feature values have been normalized to zero mean and unitary variance.

Bidirectional LSTM (BLSTM) was used to predict the emotional flow. This choice was mainly due to two reasons: to compensate possible latency when visualizing the videos for annotating the ground truth for emotion; and due to characteristics of the emotional flow, which is expected to have smooth transitions, and BLSTM may be less influenced by fluctuation on the input features.

Two experiments were performed for the fear sub-task, a trained classification model to predict the tag for every second, and also identify a segment as *fear* according to the tags

Run	Valence		Arousal	
	MSE	r	MSE	r
1	0.1021	0.1714	0.1414	**0.0870**
2	0.1036	0.1820	0.1399	-0.0181
3	**0.0924**	**0.3048**	0.1399	0.0761
4	0.0980	0.2422	**0.1396**	0.0612
5	0.0944	0.2511	0.1460	-0.0667

Table 3.7: Valence and arousal results [35].

of each second within it. Only sequences whose length is longer than a certain threshold have been kept in order to remove noise from the sequence.

Fusion methods were implemented in all experiments, such as early fusion (concatenate to a large vector, features of various modalities and sources), late fusion (output of the LSTM models is combined and used as input of the next fully-connected layer) and average fusion (average of multiple models' estimation was computed to prevent over-fitting and minimize noise).

BLSTM models were used in all runs. The first three runs are distinct in the number of the model layers, which is 4, 2 and 3 respectively. Run 4 consists of the average fusion of the first three runs. Run 5 is a late fusion of two BLSTM models, receiving in the input visual (except CNN) and audio features. Dropout with a probability of 0.5 was used in all runs to prevent over-fitting.

The results can be observed in Table 3.7 and the best are in bold. It shows that Run 3 (2-layer BLSTM model) achieved the best valence prediction result; and Runs 1 and 4 achieved the best results in correlation and MSE for arousal, respectively.

The fear sub-task performs much worse than expected and suggests that the LSTM or BLSTM models might not be suitable for the task, since the *fear* and *no-fear* classes are rather unbalanced. Therefore the runs for the fear sub-task were not been submitted.

In the approach followed in [32], the authors suggest to use single-layer and ensemble of LSTM models. The motivation was notable results obtained by using LSTMs on a similar task for emotional impact of movies proposed by *Mediaeval* in 2017.

Visual features were extracted using the VGG16 pre-trained model while audio features were extracted using the openSmile toolbox. Non overlapping sequences of 101 samples were used for feeding the LSTM.

Different LSTM model architecture variations were experimented using 12 movies as a cross-validation set. The best result was obtained by a simple single-layer LSTM with batch normalization, trained with the visual features. A LSTM with a 1D temporal convolution

	Description	IoU
Run 1	Ensemble of LSTMs + Visual and Audio features	0.06496
Run 2	Ensemble of LSTMs + Audio features	0.07507
Run 3	Ensemble of LSTMs + Visual features	0.08742
Run 4	Single-layer LSTM + Visual features	**0.11992**
Run 5	Single-layer LSTM + Audio features	0.09874

Table 3.8: Runs and fear results described by the team [32].

layer performed worse but still achieved non-zero F1 ratings, so both model architectures were considered.

The different combinations of feature sets and models depicted on Table 3.8 have been tested. Two single-layer LSTM models trained using visual features and two single-layer LSTMs trained using audio features were used in Run 1. In Run 2, visual features were used for training four LSTM models (three were single-layer LSTMs and the remaining one was a single-layer LSTM with a 1D convolutional layer attached). Run 3 is similar to Run 2, but uses audio features. Run 4 and 5 are composed by a single layer LSTM and trained using the visual and audio features, respectively.

The results for the fear sub-task can be observed in Table 3.8. The used metric was IoU. From the table, it can be observed that the ensembles performed worse than the single LSTM models. One explanation is because the predictions in the ensembles were merged by a simple average function, and not by a weighted average function. Run 4 obtained the best outcomes, which suggests that visual features may have a greater effect on detecting fear-inducing video than audio features.

3.2 Summary

This chapter described the dataset and Related work. All approaches use the *2018 MediaEva*l data from the *LIRIS-ACCEDE* dataset, except one (approach presented in [76]) that uses the *2017 MediaEval* data. The best result for *2018 MediaEval* was an IoU of 0.15750, which consisted of using audio, scene and action features. The results were not promising, since the models cannot predict the presence of fear in the films, obtaining an IoU score very low. In Chapter 4, various methods will be taken, such as splitting the data in a different manner, analyzing feature sets and models. The data will be split in the same way as in the related work (different films for training and testing), but also at the sample level (the data will be divided randomly) resulting in samples from all movies in the training and testing.

Chapter 4

Classical Classifiers and Feature Analysis

This chapter describes the experiments carried out with different classifiers for detecting *fear* inducing movie segments. The details of the experimental setup can be found in Section 4.1. The main purpose of the experiments described along this chapter is to identify combinations of feature sets and machine learning classifiers that lead to the best outcomes in the task of identifying fear inducing segments of the film. All tables presented have the best f1-score for fear prediction in bold.

In the first round of experiments, reported in Section 4.2, training and test sets are disjoint at video level, meaning that the samples in the training set come from different videos than those used in the testing set. However, this training/testing methodology lead to poor classification performance, suggesting that the the videos used in the training set are not representative of those used in the test set.

In order to tackle this problem, another approach was followed by using samples from the same videos for training and for testing. *i.e.*, training and test sets are disjoint at sample level, and thus with less restrictions than the first approach. Both the training an test sets will therefore contain samples taken from the same videos, meaning that, on the same video, a set of samples is randomly assigned to the training set while the remaining ones are assigned to the test set. The experiments that follow this approach are described in Section 4.3.

4.1 Experimental Setup

This section describes the experimental setup performed to identify feature sets that are most relevant for the fear prediction. The fear-inducing is classified in the *LIRIS-ACCEDE* database described in Chapter 3, which includes eleven sets of visual features and a single

set of audio features. The use of these feature sets is subjected to a thorough analysis along this chapter.

Several matrices have been created, where columns represent features and rows represent samples, each one corresponding to a segment with the duration of one second of film. The first segment of the first movie, for example, will be assigned to the first row of the matrix. This row contains the values corresponding to each set of features, for instance, the first 256 values correspond to the *acc*, the next 144 values the *cedd* and so on. In total, and if all feature sets are used, the number of columns is 5368, which represents the total number of features in all sets (5367) plus the last column, which represents whether or not *fear* is induced during that second of movie. This last column consists of 0's and 1's, where 0 refers to *no-fear* and 1 to *fear* inducing samples. A matrix containing data for all movies and all features has a shape of 87456 rows by 5368 columns. It is also worth to mention that the movie duration varies from film to film, and therefore the number of samples per film is not constant.

As for the machine learning classifiers used in the experiments reported in this chapter, the following have been used (most of them have been described in Section 2.1):

- Linear Discriminant Analysis.
- Decision Tree Classifier.
- K-Nearest Neighbor.
- Multi-layer Perceptron (MLP) Classifier (Neural Network).
- Gaussian Naive Bayes Classifier.
- Logistic Regression Classifier.
- Linear Support Vector Classifier.

All these classifiers were tested in order to evaluate which ones lead to the best results in the fear inducing prediction task. Their implementations are available in the *Scikit-learn* Python library.

The other main goal of this chapter is to determine which feature sets, or feature set combinations, are the most suitable for the task of predicting fear inducing movie segments. To accomplish this task, the following flow of experiments was performed:

1. Each individual set of features available in *LIRIS-ACCEDE* was, in turn, used as the classifiers input. The goal was to evaluate the discriminating power of each feature set.
2. The feature sets leading to the best results in the previous step were combined and used as the classifiers input.

3. All visual features sets were combined into a single feature vector used as input for all tested classifiers.

4. Identical to the previous step, but in this case the training set was additionally subject to a class balancing procedure .

5. Finally, all audio and visual features sets were combined.

All the experiments share the same principle: to train and predict the presence of fear-inducing video samples using the seven machine learning classifiers and variations in the feature data used as the classifiers input.

4.2 Using Different Films for Training and Testing

This section reports the experiments done using the same division of training and test sets as is in the Related Work (Section 3.1), in which the training and test sets are composed of completely different sets of videos.

The principle is that the video materials were initially split into 44 videos for training and the remaining 12 for testing. The training set contains a total of 55251 samples (one sample for each second of video), covering approximately 15 hours of video; as for the testing set, it contains 32205 samples, covering approximately 9 hours of video. The experiments described in Section 4.1 were then performed.

4.2.1 Individual feature sets

The first experiments consisted of training a set of *Scikit-learn* classification algorithms using features coming from individual sets of visual and audio features. The goal was to find out which of the feature sets are the most promising ones, when the task is to predict the temporal video intervals that induce fear into the viewer.

Since the number of samples with fear is much smaller than the number of samples without fear, accuracy is not a good evaluation metric, as seen in Section 2.6.4. In conclusion, since the dataset classes are unbalanced the F1-score was used as the assessment metric, as it uses the precision and recall metrics to compute the final outcome. The IoU metric is also robust to class unbalancing and therefore it is also an adequate performance assessment metric for the *fear* prediction task.

The results achieved by using each individual feature set are shown in Table 4.1. Each table cell depicts two values separated by '/': the first value reflects the F1-score for predicting the *no-fear* inducing class, while the second value is the F1-score for predicting the *fear* class. Each column depicts the results for a machine learning algorithm, while

	LDA	DecisionTree	KNN	MLP	GaussianNB	LR	SVM
acc	0.974/ 0.006	0.945/ 0.083	0.963/ 0.030	0.957/ 0.036	0.399/ 0.111	0.974/ 0.000	0.974/ 0.000
cedd	0.974/ 0.006	0.939/ 0.065	0.966/ 0.030	0.967/ 0.037	0.252/ 0.099	0.974/ 0.002	0.974/ 0.000
cl	0.974/ 0.000	0.942/ 0.069	0.969/ 0.009	0.974/ 0.004	**0.828/ 0.143**	0.974/ 0.000	0.974/ 0.000
eh	0.974/ 0.000	0.947/ 0.062	0.971/ 0.013	0.955/ 0.068	**0.759/ 0.143**	0.974/ 0.000	0.974/ 0.000
fc6	0.968/ 0.052	0.951/ 0.068	0.965/ 0.065	0.968/ 0.077	**0.617/ 0.135**	0.968/ 0.048	0.953/ 0.073
fcth	0.974/ 0.000	0.941/ 0.080	0.967/ 0.057	0.968/ 0.028	0.084/ 0.098	0.974/ 0.000	0.974/ 0.000
gabor	0.974/ 0.000	0.950/ 0.055	0.973/ 0.006	0.974/ 0.000	0.552/ 0.118	0.974/ 0.000	0.974/ 0.000
jcd	0.974/ 0.004	0.951/ 0.081	0.966/ 0.047	0.963/ 0.059	0.219/ 0.098	0.974/ 0.004	0.974/ 0.000
lbp	0.974/ 0.008	0.944/ 0.066	0.968/ 0.013	0.969/ 0.020	0.044/ 0.095	0.974/ 0.001	0.974/ 0.000
sc	0.973/ 0.002	0.948/ 0.060	0.970/ 0.015	0.965/ 0.036	**0.928/ 0.185**	0.974/ 0.000	0.974/ 0.000
tamura	0.974/ 0.000	0.949/ 0.085	0.968/ 0.006	0.974/ 0.007	0.974/ 0.000	0.974/ 0.000	0.974/ 0.000
audio	0.966/ 0.110	0.951/ 0.089	0.973/ 0.027	0.968/ 0.057	0.662/ 0.120	0.974/ 0.005	0.925/ 0.102

Table 4.1: *F*1-score (*no-fear*/ *fear*) achieved for each feature set.

Metrics	LDA	DecisionTree	KNN	MLP	GaussianNB	LR	SVM
F1-score	0.973/ 0.002	0.948/ 0.060	0.970/ 0.015	0.965/ 0.036	**0.928/ 0.185**	0.974/ 0.000	0.974/ 0.000
IoU	0.001	0.031	0.007	0.018	**0.102**	0.000	0.000

Table 4.2: *Sc* features set results.

each row depicts the results achieved by using the corresponding set of features in the *LIRIS-ACCEDE* dataset.

By analyzing Table 4.1, it can be observed that the feature sets leading to best results were *cl*, *eh*, *fc6* and *sc*. These feature sets represent the color of the image, the edge types in an image, values extracted using VGG16 CNN, and the color signatures, respectively. The best results regarding the detection of the *fear* class were achieved by the Gaussian Naive Bayes classifier. On the other hand, the Logistic Regression and SVM classifiers obtained the worst results regarding fear detection, with null F1-scores for nearly all feature sets, suggesting that these algorithms always classify all samples as *no-fear* inducing ones.

The time to train and predict all these models for each feature sets was high. A total of 84 models were trained and tested (7 algorithms $\times$ 12 feature sets) which took about 7 hours to complete. The model showing the higher F1-scores for the *fear* class, GaussianNB, was very fast in the training and the testing, taking only a few seconds. On the other hand, the KNN and MLP models took much longer. For instance, the KNN algorithm took about 30 minutes to train when using *fc6* feature set (feature set with the highest number of features).

The best *fear* detection results were achieved using the *sc* feature set. Table 4.2 described the results achieved using all tested classifiers. Besides the F1-score, the Intersection over Union is also depicted for a better analysis.

The best results were achieved by the Gaussian Naive Bayes algorithm, which obtained 18.5% of F1-score when predicting *fear*. It was able to correctly predict 485 out of 1606

		Predicted label	
		no-fear	*fear*
Actual label	*no-fear*	27438	3161
	fear	1121	485

Table 4.3: GaussianNB confusion matrix of *sc* feature set.

	LDA	**DecisionTree**	**KNN**	**MLP**	**GaussianNB**	**LR**	**SVM**
F1-score	0.968/ 0.045	0.952/ 0.094	0.964/ 0.073	0.966/ 0.038	**0.624/ 0.137**	0.967/ 0.040	0.951/ 0.055
IoU	0.023	0.049	0.038	0.020	**0.073**	0.020	0.029

Table 4.4: Best feature sets combined.

fear samples, and 3161 *no-fear* samples were mispredicted as *fear*, as seen in the Table 4.3. In contrast to the other strategies, all the remaining algorithms perform worse when predicting the *fear* class.

4.2.2 Best feature sets combined

The best feature sets, *sc*, *cl*, *eh* and *fc6*, were combined in order to check if better results are obtained by using a larger amount of features for classification. The results from the different algorithms are shown in Table 4.4.

By analyzing Table 4.4 it turns out that there is not much difference compared to the Table 4.1 with the individual features. The Gaussian Naive Bayes slightly decreases the *fear* prediction results. On the other hand, the *no-fear* prediction decreases even further, hitting almost fifty percent, which is not desirable. GaussianNB manages to correctly predict *fear* 1336 times, but in total predicts 17936 times the tag *fear*, leading to a large amount of false positives. The tag *no-fear* was predicted correctly 13999 times and badly 16600 times. This data can also be observed in Table 4.5, which represents the model's confusion matrix. The best Intersection over Union value is in the GaussianNB algorithm.

In terms of training and test execution time, all models take a total of 4 hours and 26 minutes to train and test. It is worth to mention that the KNN algorithm took about 3 hours and 30 minutes to run and perform predictions, making it the worst one regarding the execution time.

		Predicted label	
		no-fear	*fear*
Actual label	*no-fear*	13999	16600
	fear	270	1336

Table 4.5: GaussianNB confusion matrix of best feature set combined.

	LDA	DecisionTree	KNN	MLP	GaussianNB	LR	SVM
F1-score	0.968/ 0.034	0.956/ 0.076	0.963/ 0.063	0.963/ 0.033	**0.546/ 0.126**	0.971/ 0.01	0.954/ 0.040
IoU	0.017	0.039	0.033	0.017	**0.067**	0.005	0.020

Table 4.6: All visual features combined.

4.2.3 All visual feature sets combined

To further analyze the performance, all visual feature sets were merged and used for training and testing. As can be observed from Table 4.6, the achieved results were a bit worse than those shown in the previous experiments.

By comparing Tables 4.6 and 4.4 it can be observed that certain results were very similar, such as those for Decision Tree and KNN classifiers, while others were substantially worse. The Gaussian algorithm again decreased 12.5% its performance regarding the prediction of *no-fear* and 8.03% for the *fear* class.

It can thus be concluded that the use of the *sc* feature set alone as shown the best results so far. Furthermore, the use of a single feature set requires less time for training, and memory consumption is minimized. Since the results are getting worse as more features are used, it can be speculated that, as more feature sets are used, the more noticeable becomes the overfitting to the training set. Remember that samples used for training come from different films than those used for testing, which suggests that the training set is not representative of the film universe.

The total time required for training, using all visual features combined, was identical to the experiment described in Section 4.2.2, taking a total time of about 4 hours.

4.2.4 Class balancing with all visual features sets

Class balancing with all visual features sets was performed because of the 87456 samples in the dataset, 83759 samples were labeled as *no-fear* (95.8%), and 3697 samples as *fear* (4.2%). Since the *no-fear* class has a much a larger number of samples when compared with the *fear* class, the classes are highly unbalanced. This fact contribute to the reason why several experiments produced a very small amount of classifications that fall in the *fear* class. Therefore, an experiment was carried out by balancing the dataset classes.

Only the training data was balanced. This was achieved by removing random samples with the tag *no-fear* from the training set. In the end, the number of *fear* samples is the same as the number of *no-fear* samples, resulting in a much smaller training set with only 4182 samples. Similarly to the previous experiment, all the visual feature sets were combined.

The results are depicted in Table 4.7. When compared with the results of the previous experiment, the prediction of *fear* generally increases, but at the cost of a substantial

	LDA	DecisionTree	KNN	MLP	GaussianNB	LR	SVM
F1-score	0.730/ 0.097	0.807/ 0.103	**0.763/ 0.133**	0.882/ 0.110	0.549/ 0.127	0.889/ 0.089	0.864/ 0.089
IoU	0.051	0.054	**0.071**	0.058	0.068	0.047	0.046

Table 4.7: All visual features combined and classes were balanced.

	LDA	DecisionTree	KNN	MLP	GaussianNB	LR	SVM
F1-score	0.969/ 0.059	0.95/ 0.084	0.972/ 0.039	0.971/ 0.032	**0.679/ 0.152**	0.974/ 0.03	0.964/ 0.049
IoU	0.030	0.044	0.020	0.016	**0.082**	0.015	0.025

Table 4.8: Visual and audio features combined.

decrease in the prediction of *no-fear*. Compared to Table 4.4, it can be seen that all IoU metric values increased except the Gaussian model, for example the KNN increases 46.48% and Gaussian decreases 6.85%, but their values continue to be too low. In addition, the best F1-score result remains with *sc* feature set.

This approach took the least time to train all models (about 20 minutes), which is due to the fact that the size of the training data is much smaller than the one used in the previous experiments. In this case, all models were trained with 4182 samples rather than the 55251 samples.

4.2.5 All features sets combined

In this experiment, all available *LIRIS-ACCEDE* visual and audio feature sets have been used. When combined, these feature sets correspond to a number of 6951 features per sample. The features sets included in this section are the visual, equivalent to Section 4.2.3, plus the audio, resulting in a training and testing set represented in *pandas* (a Python library) by 6951 columns (total number of features).

Table 4.8 synthesizes the results of this experiment. Comparing the results with the results from the balanced class experiment in Table 4.7, it can be concluded that the Gaussian Naive Bayes algorithm is the only one that achieved better results. Nevertheless, results from this algorithm using all available feature sets were worse than those achieved using the *sc* feature set alone.

This approach was expected to take longer due to having a larger amount of features but in the end it only took about 2 hours and 28 minutes for training and testing, which means that the training algorithms converged faster. The KNN algorithm took about 32 minutes, instead of hours as happened before. In this situation, the MLP was the model that took the longest during the training, taking 44 minutes to complete.

Method	F1-score	IoU	Time to train and test (s)
Individual features - *sc*	0.185	0.1017	0,095
All features combined	0.152	0.0820	12,7167
Individual features - *cl*	0.143	0.0771	0,0535
Individual features - *eh*	0.143	0.0772	0,1224
Best features combined	0.137	0.0734	8,7499

Table 4.9: Best results for Gaussian Naive Bayes Classifier.

4.2.6 Summary

By comparing the results achieved in the experiments described from Section 4.2.1 to 4.2.5, the best ones where achieved by using the *sc* feature set alone. Table 4.9 synthesizes the performance achieved for the most promising five setups, from best to the worst. All these methods obtained the best result by using a Gaussian Naive Bayes classifier.

Inspecting Table 4.9, the global outcome is poor, at 18.5% of F1-score by predicting the label *fear*. The time required to train and test are very low, but in the end, the best method (Individual features - *sc*) obtained one of the shortest times.

The experiments suggest dataset limitations. The dataset classes are very unbalanced, which causes that most models are prone to forecasting the *no-fear* class since this class represents about 95.8% of samples used in training. On the other hand, balancing the data by removing elements of the most numerous class lead to a rather small training set.

The experiments also showed that the use of a larger amount features did not translate into better performance. This fact may be due to overfitting problems that become more evident as more features are used.

4.3 Using Samples From Same Films for Training and Testing

The experiments done in Section 4.2 showed that the data used is cleary insufficient which makes the models do not generalize well. In order to overcome this problem, this section describes a new set of experiments at the sample level. A portion of the movie samples are used for training, and another for testing, and this is done by randomly split the data. The algorithm's training and test processes will therefore access samples coming from the full films dataset.

The main database containing a total of 87456 samples was randomly split into 20% of all samples for testing (17492 samples) and 80% for training (69964 samples).

The flow of experiments previously performed in Section 4.2 was also performed for this new dataset division, in order to find out which algorithms and feature sets lead to the

	LDA	DecisionTree	KNN	MLP	GaussianNB	LR	SVM
acc	0.976/ 0.136	0.974/ 0.424	**0.986/ 0.637**	**0.981/ 0.539**	0.383/ 0.098	0.978/ 0.028	0.977/ 0.064
cedd	0.978/ 0.013	0.976/ 0.467	**0.985/ 0.584**	**0.982/ 0.507**	0.338/ 0.093	0.978/ 0.013	0.978/ 0.000
cl	0.978/ 0.000	0.972/ 0.411	0.983/ 0.497	0.979/ 0.093	0.821/ 0.127	0.978/ 0.000	0.978/ 0.000
eh	0.978/ 0.000	0.959/ 0.178	0.983/ 0.468	0.970/ 0.271	0.804/ 0.114	0.978/ 0.000	0.978/ 0.000
fc6	0.977/ 0.405	0.964/ 0.233	**0.986/ 0.619**	**0.982/ 0.570**	0.709/ 0.122	0.979/ 0.359	0.972/ 0.379
fcth	0.978/ 0.000	**0.977/ 0.500**	**0.985/ 0.568**	0.982/ 0.451	0.132/ 0.087	0.978/ 0.000	0.978/ 0.000
gabor	0.978/ 0.000	0.959/ 0.096	0.977/ 0.005	0.978/ 0.000	0.723/ 0.110	0.978/ 0.000	0.978/ 0.000
jcd	0.978/ 0.010	0.977/ 0.467	**0.985/ 0.599**	**0.982/ 0.550**	0.274/ 0.092	0.978/ 0.011	0.978/ 0.000
lbp	0.978/ 0.016	0.970/ 0.335	0.982/ 0.391	0.979/ 0.180	0.126/ 0.085	0.978/ 0.011	0.978/ 0.008
sc	0.976/ 0.092	0.972/ 0.364	0.983/ 0.498	0.979/ 0.420	0.898/ 0.149	0.978/ 0.000	0.978/ 0.000
tamura	0.978/ 0.000	0.961/ 0.185	0.980/ 0.255	0.978/ 0.005	0.951/ 0.055	0.978/ 0.000	0.978/ 0.000
audio	0.978/ 0.342	0.976/ 0.479	0.979/ 0.188	0.982/ 0.521	0.619/ 0.107	0.978/ 0.042	0.853/ 0.107

Table 4.10: Individual features (F1-score).

		Predicted label	
		no-fear	*fear*
Actual label	*no-fear*	16633	111
	fear	347	401

Table 4.11: KNN confusion matrix of *acc* feature set.

best results. The following subsections depict the experimental results.

4.3.1 Individual feature sets

Similar steps to those described in sub-Section 4.2.1 were followed and, for each feature set, the same machine learning algorithms were used for the predictions. The results obtained are shown in Table 4.10. Similarly to what has been done before, the F1-scores were used for performance assessment.

By observing Table 4.10, it can be seen that the results obtained are much better than those depicted in Table 4.1. With the new training/testing set division, the best F1-score regarding the prediction of *fear* is much higher. Training and testing with samples coming from the same films lead to F1-scores predicting *fear* in the sixties per cent, for the KNN and MLP Classifiers, with the *acc*, *cedd*, *fc6*, *fcth* and *jcd* feature sets being the most promising ones. An F1-score of 63.7% for the *fear* class was the best result achieved, using the KNN model with the *acc* feature set, representing the image color correlogram.

The test data contains 748 samples of the *fear* class and 16744 of the *no-fear* class. The algorithm achieving the highest F1-score in *fear* prediction – KNN –, managed to correctly predict 401 samples of *fear* and 16633 of *no-fear*, as can be observed in the confusion matrix represented in Table 4.11.

When compared to the results presented in Section 4.2, the reason for this shift to higher F1-scores is due to the new training/ test set division. In this experiment, the training

	LDA	DecisionTree	KNN	MLP	GaussianNB	LR	SVM
F1-score	0.978/ 0.472	0.967/ 0.327	**0.987/ 0.65**	0.983/ 0.609	0.575/ 0.114	0.98/ 0.431	0.961/ 0.377
IoU	0.309	0.196	**0.481**	0.438	0.06	0.275	0.232

Table 4.12: Best features combined.

		Predicted label	
		no-fear	*fear*
Actual label	*no-fear*	16625	119
	fear	331	417

Table 4.13: KNN confusion matrix of best feature sets combined.

set contains samples coming from all the films available in the dataset, and therefore it is a richer set. On the other hand, the test set, although consisting of samples different from those used for training set, contains samples that are expected to be more correlated with those used for training, than in the situation that they were coming from different films.

Similarly to Section 4.2.1, this set of experiments demanded a long time to train and test. It took about 9 hours and 54 minutes.

4.3.2 Best feature sets combined

The best feature sets, with the highest F1-score, in Table 4.10, were combined to try to achieve better results. Five feature sets were selected – *acc*, *cedd*, *fc6*, *fcth* and *jcd* –, resulting in 5368 features for each second. The results with all these feature sets combined are depicted in Table 4.12. This method takes about 4 hours to train and test all algorithms.

KNN again was the model with the best F1-score for predicting *fear*, achieving 65% of the *fear* class and 98.7% of the *no-fear*. Compared to Table 4.10, *fear* and *no-fear* class are improved by 0.013 and 0.001, respectively, of F1-score. Also, *MLPClassifier* has a F1-score value of 50 percent in the Section 4.3.1, and it shifts to 60 percent, achieving better results. Comparing this Intersection over Union metric with the previous *best features combined,* Section (4.2.2) using different movies to train and test, it is possible to see that the value increases from 0.073 to 0.481. In addition, the best algorithm has improved, before it was GaussianNB, and now it's KNN with 65 percent *fear* estimation, increasing 46.5%. The GaussianNB algorithm has poorer result in this case, comparing with all the algorithms.

It is possible to see that the model KNN was able to forecast 417 of the 748 *fear* tags and 16625 of the 16744 *no-fear* tags by examining the confusion matrix in Table 4.13.

4.3.3 All visual feature sets combined

All the eleven visual feature sets were combined in one dataframe, with the help of *pandas*, and then all individual algorithms are trained and tested. In total, it takes approximately 7

	LDA	DecisionTree	KNN	MLP	GaussianNB	LR	SVM
F1-score	0.979/ 0.493	0.969/ 0.351	0.986/ 0.631	**0.985/ 0.64**	0.672/ 0.125	0.979/ 0.313	0.979/ 0.455
IoU	0.327	0.213	0.461	**0.471**	0.066	0.186	0.294

Table 4.14: All visual features combined.

	LDA	DecisionTree	KNN	MLP	GaussianNB	LR	SVM
F1-score	0.768/ 0.143	0.829/ 0.182	0.846/ 0.237	**0.91/ 0.316**	0.701/ 0.13	0.883/ 0.261	0.794/ 0.194
IoU	0.077	0.1	0.134	**0.188**	0.069	0.15	0.107

Table 4.15: Results of balanced classes.

hours and 53 minutes to train and test all the algorithms, and in the end, the results were a little higher in some cases and in others remained the same.

Table 4.14 represents the F1-score and IoU results for each tested ML algorithm. For this experiment, the MLP classifier achieved the best outcome. Comparing with the results depicted in Section 4.3.2, it can be observed that the use of all feature sets is leads to similar results, going from 0.65 to 0.64 for the *fear* class prediction. The IoU is also very similar, slightly decreasing from 0.481 to 0.471. The results concerning the prediction of the *no-fear* class are even more similar, with F1-score differences of about 0.002.

4.3.4 Class balancing with visual features sets

This method consists of balancing the classes of *fear* and *no-fear*. The *no-fear* class was randomly reduced to the same number of samples from the *fear* class, as in Section 4.2.4, resulting in a size of 5898 seconds for training, rather than 69964. It took less time to train and test all algorithms, approximately 16 minutes, since this method had fewer samples in training.

The results were not the best with a class balancing, as can be seen in Table 4.15. The best result is 38% well below the previous ones, so a very poor result is obtained by this approach. This may have happened because important data was removed. Analyzing the confusion matrix of the model MLP, in Table 4.16, it is possible to verify that the *fear* prediction has 2669 false positives (*no-fear* tag has been wrongly predicted), and this happens because 64066 seconds were removed with the *no-fear* tag.

		Predicted label	
		no-fear	*fear*
Actual label	*no-fear*	14075	2669
	fear	107	641

Table 4.16: MLP confusion matrix of balanced classes.

	LDA	DecisionTree	KNN	MLP	GaussianNB	LR	SVM
F1-score	0.983/ 0.619	0.973/ 0.436	0.981/ 0.32	**0.988/ 0.725**	0.729/ 0.137	0.979/ 0.16	0.981/ 0.397
IoU	0.448	0.279	0.191	**0.569**	0.073	0.087	0.248

Table 4.17: All features sets combined results.

		Predicted label	
		no-fear	*fear*
Actual label	*no-fear*	16546	198
	fear	210	538

Table 4.18: MLP confusion matrix all features sets combined.

4.3.5 All features sets combined

In this approach all the features were combined. The differences of this approach with Section 4.3.3 is the addition of the audio feature. As is explained at the beginning of the Section 4.3, this data is divided randomly. To verify the results, the models were again trained. Impressively, the training and testing period was shorter than in previous methods, taking 2 hours and 48 minutes because what was observed was that with more features, more time was needed to train and test all algorithms (happen the same in Section 4.2.5). The results of these trained algorithms are shown in the Table 4.17.

The best result so far, hitting 72.5% of F1-score in the prediction of *fear*, is obtained by the MLP classifier. Also, the F1-score of the *no-fear* class increases 0.1%, achieving 98.8% of F1-score. An inspection of the confusion matrix in Table 4.18 reveals that the *fear* class was well classified 538 times and badly 198, and the *no-fear* class was correctly predicted 16546 times.

4.3.6 Summary

Comparing all the approaches described along this section, it is possible to conclude that the integration of all audio and visual features produced the better outcomes. The five strategies experimented leading to the best results are synthesized in Table 4.19.

The best results were achieved by the MLP classifier, which achieved a F1-score of

Method	Algorithm	F1-score	IoU	Time to train and test (min)
All features combined	MLP	0.725	0.569	51.36
Best features combined	KNN	0.650	0.481	172.95
All visual features combined	MLP	0.640	0.471	307.54
Individual features - acc	KNN	0.637	0.467	6.75
All visual features combined	KNN	0.631	0.461	105.80

Table 4.19: Best results.

Figure 4.1: Three frames from the movie 0.

Figure 4.2: Second segment of the movie 0.

72.5% predicting the *fear* tag and 99.8% predicting the *no-fear*, when all the available features (visual and audio) were combined. This algorithm feature set combination was also the one that took less processing time to train and test, comparing to *Best features combined* and *All visual features combined*.

4.4 Fear-inducing Classifications

In this section some of the movies were manually watched to check if they were good movies (based on real life and not on nonsense material, have good resolution, etc) and if the tag *fear* was correctly placed. A function was created to generate subtitles in the movies. The purpose of this function is to warn pictures that include fear in the numerous videos. A subtitle that says "fear" has been applied to the videos. This makes it easy to manually check if the fear tag was well put and if the dataset was done well.

Figure 4.1 shows three frames from the movie *MEDIAEVAL18 00* and reveal that the first two frames (explosion and body) convey fear, but are not identified as such, and the third frame, which contains a person's body, is correctly tagged as fear.

Figure 4.2 shows an image that transmits fear, but the audio contains a piece of happy music, so the audio does not correspond with the current action, thus the audio, in this case, was misleading to recognize fear.

The *fear* tag in the dataset contains scenes like a fight and with suspense, images with

Figure 4.3: Segment of the movie 16.

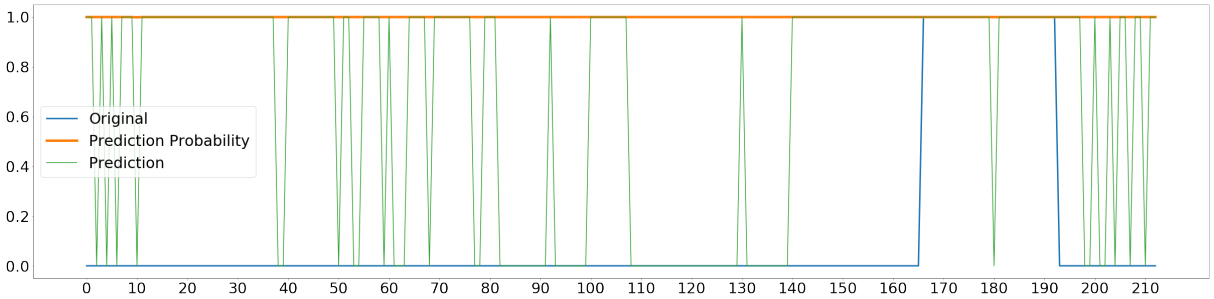

Figure 4.4: Gaussian predictions for visual features.

blood, pistols, but there are some cases where this doesn't happen. This is illustrated in Figure 4.3, which shows a person placed in a bag with a man carrying a gun in his pocket, and this frame has no annotation of *fear*.

The conclusion is that the *fear* annotations are not consistent. In some cases, the *fear* tag is well-associated, and some are not, plus there are some cases of fear that do not exist. Feeling of fear can also vary person to person, so it's hard to tell for certain whether or not a picture involves fear.

To examine what the best model in Section 4.2 predicts in each second and the confidence of the predicitons, a graph has been made for each movie. This graph represents, in each second, the real tag *fear* or *no-fear* (1 or 0 in the axis y), the prediction that the model qualifies (green line) and the confidence in each prediction (orange). This likelihood is the assurance that the model has, for example, if the model says that in the first second the probability to predict *fear* is 1, this means that the model has 100% confidence that fear occurs at that time. Figure 4.4 corresponds to movie 54 in the dataset of *LIRIS-ACCEDE*. The x axis represents each second of the movie, in this case the movie has 213 seconds, and these predictions were made using visual features only.

Analyzing Figure 4.4 is it possible to see that the model outputs a lot of random predictions at the beginning of the movie, the green line goes up and down (classify as *fear* and *no-fear*) at a time when there is no fear conveyed. In all of these predictions, the model has a 100% predictability, which is definitely not a positive indication. Looking at these

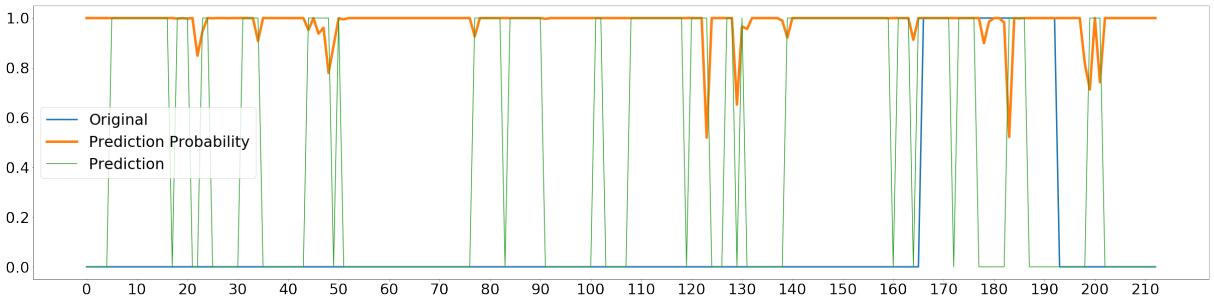

Figure 4.5: Audio features predictions.

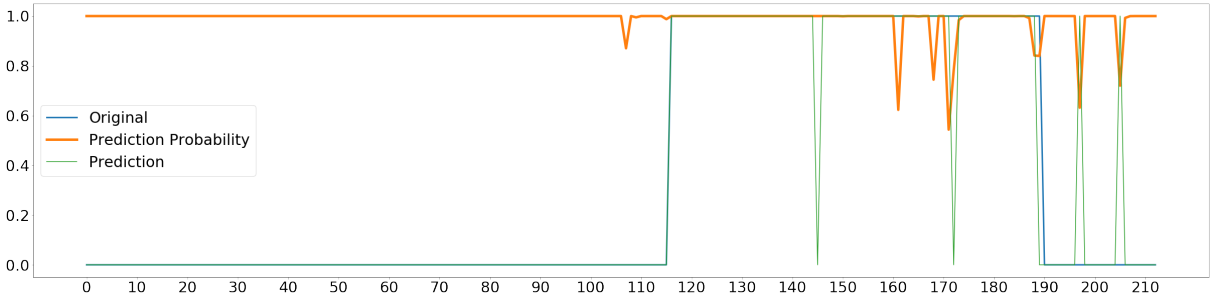

Figure 4.6: Predictions using the best method.

results, it is possible to conclude that this is not a good model since it performs a lot of *fear* predictions with only one second. In one second exists fear, and the next one doesn't. This doesn't occur in movies, if the movies transmit fear, that fear will be transmitted in the next seconds and not in one unique second. In this movie, it is possible to see that exists fear in the seconds 165 to the 193, and in all graph, there's no peak of fear in a unique second.

With the audio features, represented in Figure 4.5, it's possible to see the same thing, but with these features, the predictability isn't necessarily 100%.

The same methods for the best model in Section 4.3 have been implemented and the graph is represented in Figure 4.6. The model was trained with a random split of the data and certain movies were chosen to perform prediction, in summary, the training is randomly separated but the test is not.

Figure 4.6 represents the movie 55 with the real and predicted tag, and the probability of certain in each prediction, and as expected, forecasts are much better at predicting nearly all correctly, and in certain situations, if the model is inaccurate, the probability of confidence that it has is poor, having around 60% certainty of the prediction.

If improvements were made to the model predictions by changing isolated predictions, better results could be obtained because there is no likelihood of *fear* in a single second or vice versa. An example of change is to change the model's prediction, for example, *fear* in a single second, to *no-fear*.

4.5 Summary

Some conclusions can be drawn by comparing the results obtained with those of related works. The best result achieved in related works using the same data as in this chapter (*MediaEval* 2018) was an IoU of 0.15750. This result, proposed in [73], uses audio, scene and action features and divides the data the same method as Section 4.3, different movies to train and test. Using this type of division it was possible to obtain a result of 0.1017 IoU utilizing only the feature *sc.* Both results were not positive, as the model cannot predict fear.

When comparing the use of training/test set division at film level with division at sample level (4.2 vs 4.3), the latter achieved better results. F1-scores improved from 0.185 to 0.725. When training and testing with distinct films there are far too many classification errors, meaning that learning based on a set of films assigned to the training set did not generalize for different films, and this is a very bad result which it is possible to say that the model can't predict *fear*. On the other hand, the predictions were much more accurate when the ML algorithms were trained with about 80% of the samples coming from all the videos present in the dataset, meaning that generalization is achieved for different moments of the films. These observations suggest that the *LIRIS-ACCEDE* database may not be sufficiently representative of the films universe. A larger dataset covering more film variations should help to improve the learning processes.

Chapter 5

Deep Learning Classification

In this chapter, the Python library called *Tensorflow* is going to be used in order to try to achieve improved results compared to Chapter 4. First, a baseline Neural Network is created to verify what results are obtained, and then a deeper NN is deployed. Lastly, an RNN is generated using the LSTM layers.

5.1 Baseline Neural Network

A Neural Network based on the [6] methodology work, containing three hidden layers of 64, 32, 32 neurons with an activation function called Rectified Linear Unit function, and a learning rate equal to 0.001 was deployed. In this case, the NN contains four dense layers and one dropout. The output, last dense layer, is composed of a dense layer with an activation function sigmoid, which returns only values between 0 and 1. The optimizer used is the Adam and has a loss function called binary cross-entropy. The dropout layer was used with a factor of 0.5, meaning that 50% of the connections will be dropped and whatever enters the dropdown will leave with half of the connections. The structure of this NN can be seen in Figure 5.1.

```
Layer (type)       Output Shape              Param #
=================================================================
Dense 1             (None, 64)                343552
_________________________________________________________________
Dense 2             (None, 32)                2080
_________________________________________________________________
Dense 3             (None, 32)                1056
_________________________________________________________________
Dropout             (None, 32)                0
_________________________________________________________________
Dense 4             (None, 1)                 33
=================================================================
Total params: 346,721
```

Figure 5.1: Model created using TensorFlow.

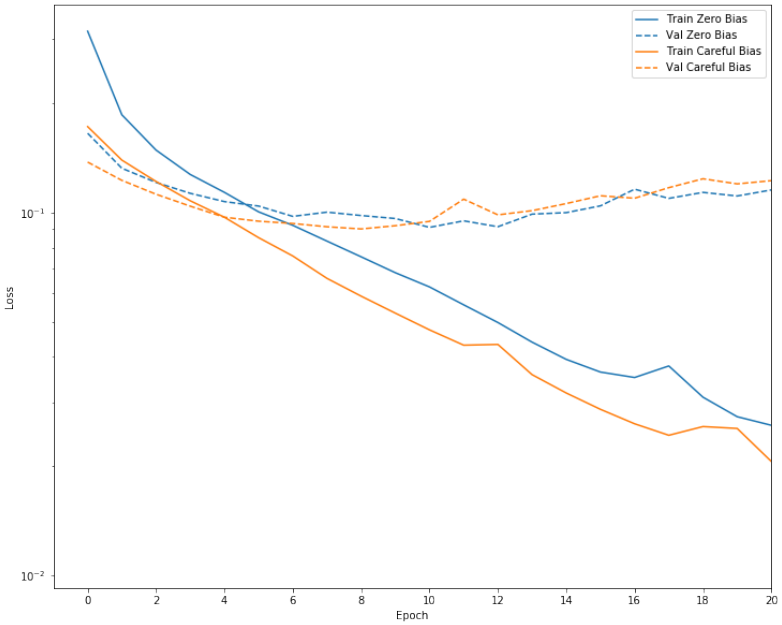

Figure 5.2: Number of epochs by loss.

A tutorial/code made by TensorFlow [67] was used to classify the imbalanced data. The *LIRIS-ACCEDE* dataset is imbalanced since the *fear* tag is much less frequent than the *no-fear* tag, and in total there are 87456 seconds in which only 4.23% of all data contain the tag *fear*. In this case the data used is the eleven visual features sets (the data is smaller, comparing to all features sets, so the training and testing period is shorter, which makes it easier to check whether the results are positive or bad), and samples were randomly split for training and for testing (80% training / 20% testing).

In order not to waste the model's first epochs discovering that the *fear* tag is unlikely, the bias was changed using the eq. (5.1), retrieved from [67].

$$bias = -log_e \frac{samples\, fear}{samples\, no\, fear} \tag{5.1}$$

To show the benefit of bias shifting, the plot represented in Figure 5.2, depicts the train and validation loss along 20 epochs, with and without the bias shifting using the data and model described previously.

It is possible to see in Figure 5.2 that the change in bias improves a little in the training of the model, but the loss of the validation data worsened in the ninth epoch, and in the twentieth epoch the loss became almost the same as the model without the bias. Cross-entropy loss is used because there are only two mark classes, 0 and 1, in this case, 0 is *no-fear* and 1 is *fear*.

A model was trained. This baseline model contains the calculated bias, 100 epochs with an early stopping monitoring the recall and a batch size of 4096.

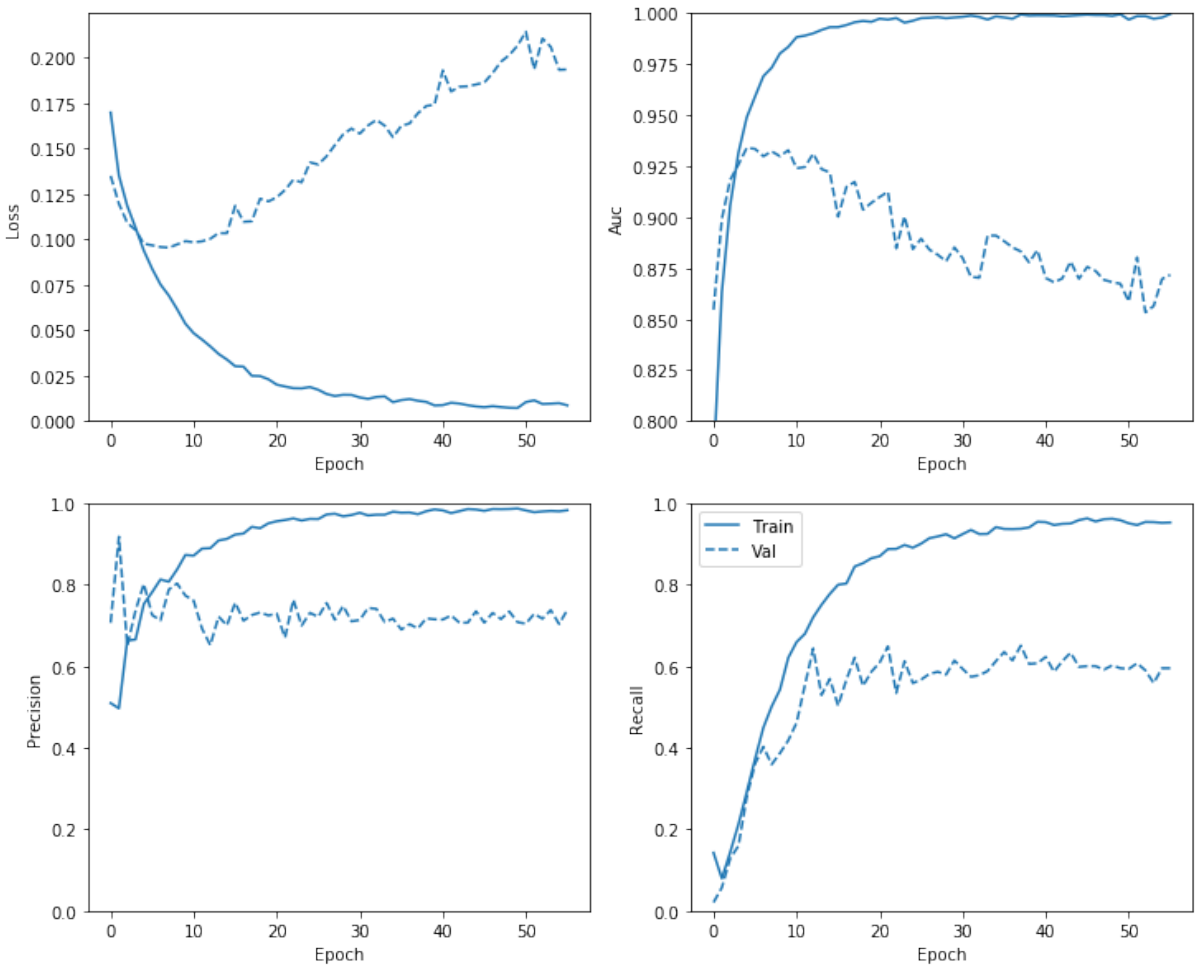

Figure 5.3: Variation of the loss, accuracy, precision and recall.

		Predicted label	
		no-fear	*fear*
Actual label	*no-fear*	16586	180
	fear	273	453

Table 5.1: Confusion matrix.

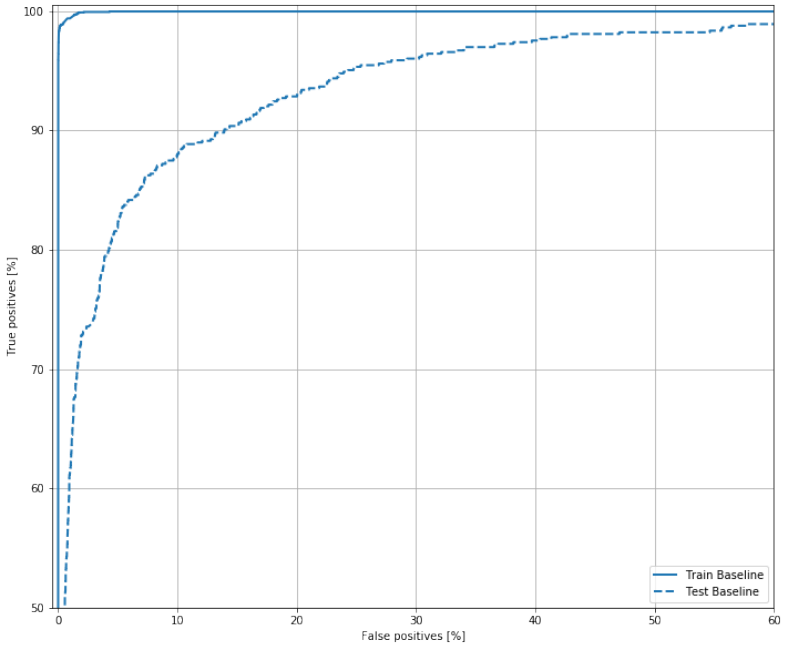

Figure 5.4: ROC curve.

Analyzing Figure 5.3, the train line in recall doesn't change much, from epoch 30, in the training curve, and the validation curve stays in the 0.6 recall. Also, the loss in the validation curve increases, which means that the longer the model is trained, the more loss exists in the validation, in fact, this can mean that the model is over-fitting. This NN starts to learn patterns that are appropriate to the training set and not good for generalization. This leads the validation set to predict wrong, amplifying the loss [1]. On the other hand, the loss in the training curve decrease. Accuracy, precision and recall make it easier to see and provide other evidence that the model is over-fitting since the curve is nearly one. This can occur because the data with the tag *no-fear* is much higher than the *fear* tag.

The testing with the baseline model was then used to perform predictions in the remaining data. This model obtained an F1-score by predicting *fear* of 67.87% and *no-fear* 98.57%. Table 5.1 illustrates the confusion matrix of the model.

A ROC (Receiver Operating Characteristics) graph was implemented to see the performance of the model because the ROC curve tells how the model classifies the *fear* tag as *fear* and the *no-fear* tag as *no-fear*.

Analyzing the results obtained from the model, in Figure 5.4, it is possible to see that the AUC in the validation is between 0.5 and 1 so that the algorithm can differentiate

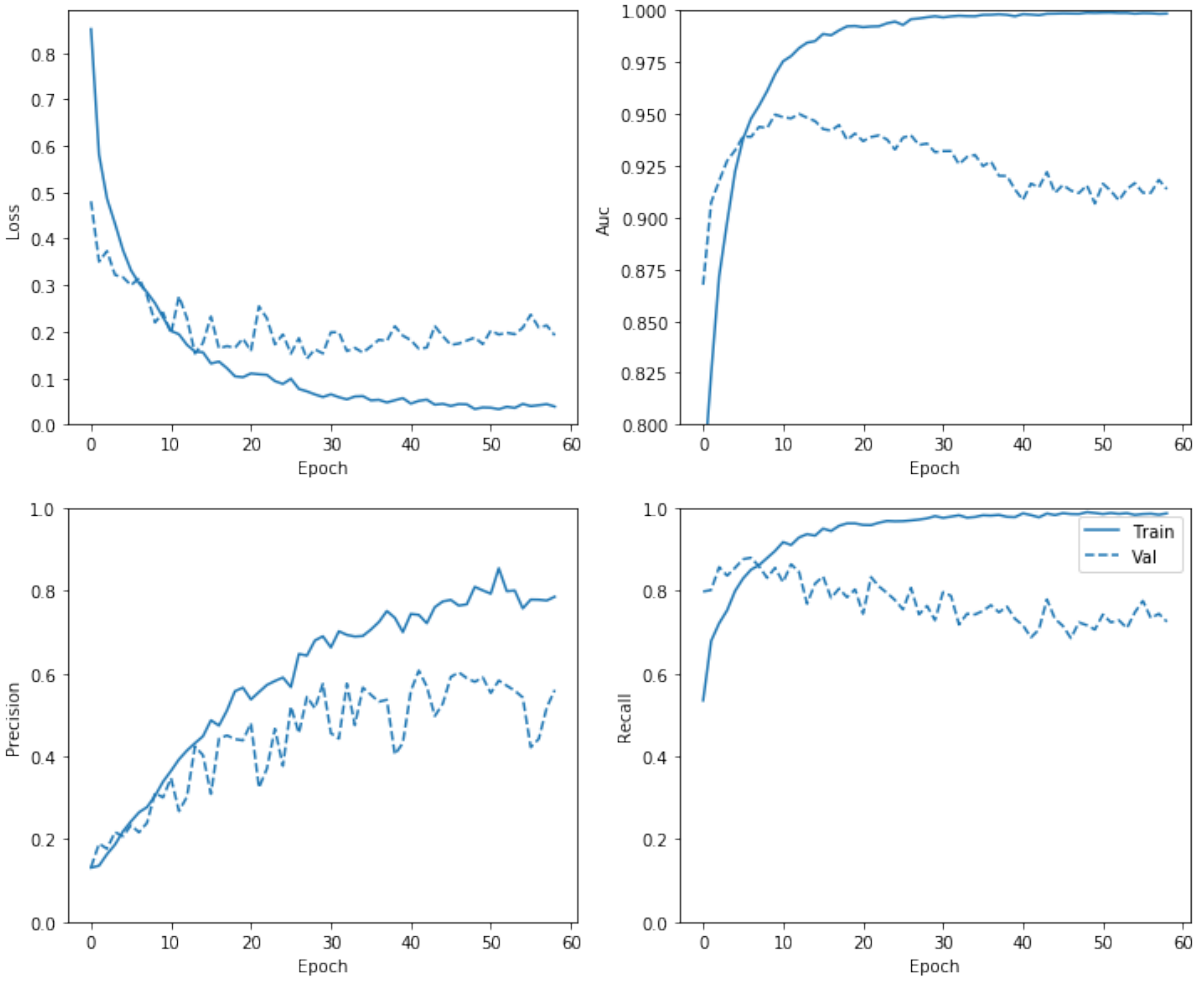

Figure 5.5: Number of epochs by the loss, accuracy, precision and recall with class weights.

between *fear* and *no-fear*. The AUC in the test is almost 1, which means that the model can distinguish almost all tags and this curve can intend to be an overfit on the train.

Then the same model but this time with different class weights calculated is trained. These weights are calculated because the class *fear* is the one that has fewer samples so needed too have more importance. This may lead the model to pay more attention to examples in the under-represented class [67]. The *no-fear* class has a weight of 0.52 and the class of *fear* is 11.83, applying eq.5.2, retrieved from [67].

$$weight = \frac{total}{2 * class\ size} \tag{5.2}$$

Analyzing the graphics in Figure 5.5 is possible to see that the loss no longer decreases. In addition, the recall increases and the precision worsens, but overfitting seems to continue to exist. The results were proximally close to the baseline model achieve an F1-score by predicting *fear* of 70.74% and *no-fear* 98.54%.

As shown in Table 5.2, representing the confusion matrix, this model can predict

		Predicted label	
		no-fear	*fear*
Actual label	*no-fear*	16399	367
	fear	198	528

Table 5.2: Model with class weights.

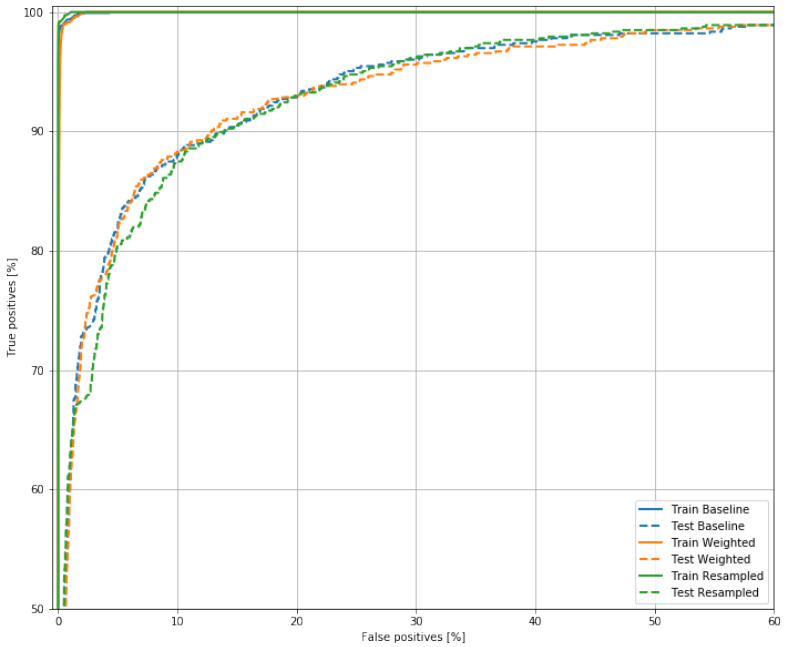

Figure 5.6: ROC with all the models.

correctly the class *fear* 528 times and 16399 times the *no-fear* class but predict incorrectly 565 times.

Lastly, a model with an oversampled data is trained. *Fear* class will be oversample so that there is more data about *fear*. The model obtain an F1-score by predicting *fear* of 68.68% and *no-fear* 98.64%.

The ROC results for all models are shown in Figure 5.6. In the end, the study of the ROC graph reveals that all the models obtains the same value of AUC in the testing, and in the training all methods obtain a value close to 1, more specifically, the AUC in three models is 0.8833, 0.9234 and 0.8798. In testing, the AUC has a value greater than 0.5, which means that the models can differentiate the classes *fear* and *no-fear*.

Also, the data about the audio features were used. The same procedures were used and the best result was 67,62% of F1-score by predicting *fear*.

Even tests using different films to train and test were made. The results were poor, 8,283% in the videos features sets were the best F1-score by predicting *fear*. By forecasting *fear*, the audio features get a 9.687% of F1-score. The *sc* feature, which uses the Gaussion algorithm to achieve the best result (18,5%), generates a result of 6.904% using this NN algorithm.

	F1-score *fear* (%)	F1-score *no-fear* (%)	IoU	Time to train (min)
Baseline model	78.367	99.101	0.644	1.616
Weighted model	77.859	98.908	0.637	1.290
Oversampled model	78.358	99.073	0.644	8.084

Table 5.3: F1-score using the model with 5 hidden layers.

The audio and visual features combined were also used, as these features sets had the best result with the MLP classifier, *Scikit-learn* algorithm (72.5% F1-score). So, the Neural Network with the audio and visual features sets is deployed to see what results are obtained, whether they are better or worse. Table 5.3 represents the approaches performed following the same idea as previously.

The results in Table 5.3 produce improved results relative to the MLP *Scikit-learn* algorithm, rising by about 6% in the F1-score of *fear* and 0.3% of *no-fear*. Also, IoU goes from 0.569 to 0.644. It should be noted that the training time has also decreased from 51 minutes to approximately 2.

5.2 Deep Neural Networks

Deeper learning was performed in order to try to achieve better results. Firstly, the visual feature sets are the data utilized and the data used in this section and Section 5.1 were randomly separated so that the train and test data are different in the two sections. The network's complexity has been increased, with 10 dense layers and 9 dropouts because several experiments were done with different layers and this model was the one with the best result. Figure 5.7 reflects the new NN.

In an effort to prevent overfit, the 9 dropouts were included, but the results of the F1-score did not change too much, staying in the seventies. Using the dropouts, gazing at the same graphs that tell us the training history, helps to remove the overfit, as seen in Figure 5.8.

The experiments performed in Section 5.1 were also done with this Neural Network (train the model with a calculated bias, also with class weights, and for the last train on the oversampled data). The best result of F1-score is obtained by class weights and the value of these weights is the same as Section 5.1, so the best F1-score result is 74.22% in the *fear* class and 98.99% in the *no-fear*, increasing 3.48% and 0.45% respectively. In Table 5.4, the confusion matrix is represented and tells how this NN behaves in the prediction of *fear* and *no-fear*. It forecast *fear* 488 times out of 729 and *no-fear* 16665 times out of 16763, and misses 339 times in total.

Experiments have also been carried out for all available data, visual and audio features sets, as these are the ones that have achieved the greatest outcomes so far. Table 5.5

```
Layer (type)        Output Shape              Param #
=================================================================
Dense 1             (None, 1024)              5496832
_________________________________________________________________
Dropout 1           (None, 1024)              0
_________________________________________________________________
Dense 2             (None, 256)               262400
_________________________________________________________________
Dropout 2           (None, 256)               0
_________________________________________________________________
Dense 3             (None, 256)               65792
_________________________________________________________________
Dropout 3           (None, 256)               0
_________________________________________________________________
Dense 4             (None, 256)               65792
_________________________________________________________________
Dropout 4           (None, 256)               0
_________________________________________________________________
Dense 5             (None, 64)                16448
_________________________________________________________________
Dropout 5           (None, 64)                0
_________________________________________________________________
Dense 6             (None, 64)                4160
_________________________________________________________________
Dropout 6           (None, 64)                0
_________________________________________________________________
Dense 7             (None, 32)                2080
_________________________________________________________________
Dropout 7           (None, 32)                0
_________________________________________________________________
Dense 8             (None, 32)                1056
_________________________________________________________________
Dropout 8           (None, 32)                0
_________________________________________________________________
Dense 9             (None, 32)                1056
_________________________________________________________________
Dropout 9           (None, 32)                0
_________________________________________________________________
Dense 10            (None, 1)                 33
=================================================================
Total params: 5,915,649
```

Figure 5.7: Model with 10 Dense layers and 9 Dropouts.

		Predicted label	
		no-fear	*fear*
Actual label	*no-fear*	16665	98
	fear	241	488

Table 5.4: Confusion matrix.

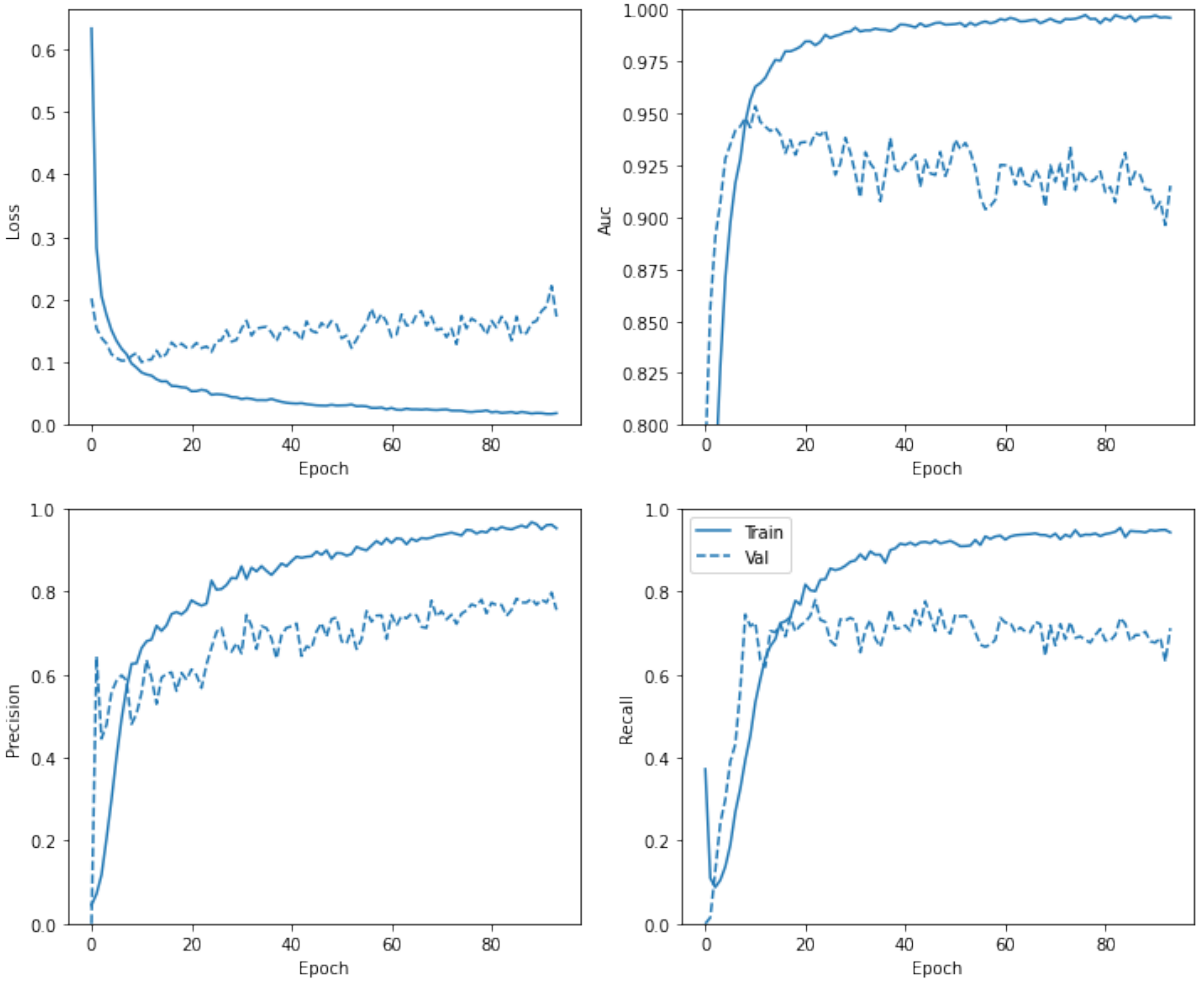

Figure 5.8: Number of epochs by the loss, accuracy, precision and recall.

	F1-score *fear* (%)	F1-score *no-fear* (%)	IoU	Time to train (min)
Baseline model	79.686	99.231	0.662	1.943
Weighted model	82.694	99.289	0.705	1.787
Oversampled model	84.014	99.336	0.724	18.143

Table 5.5: F1-score using the model with 10 Dense layers and 9 Dropouts.

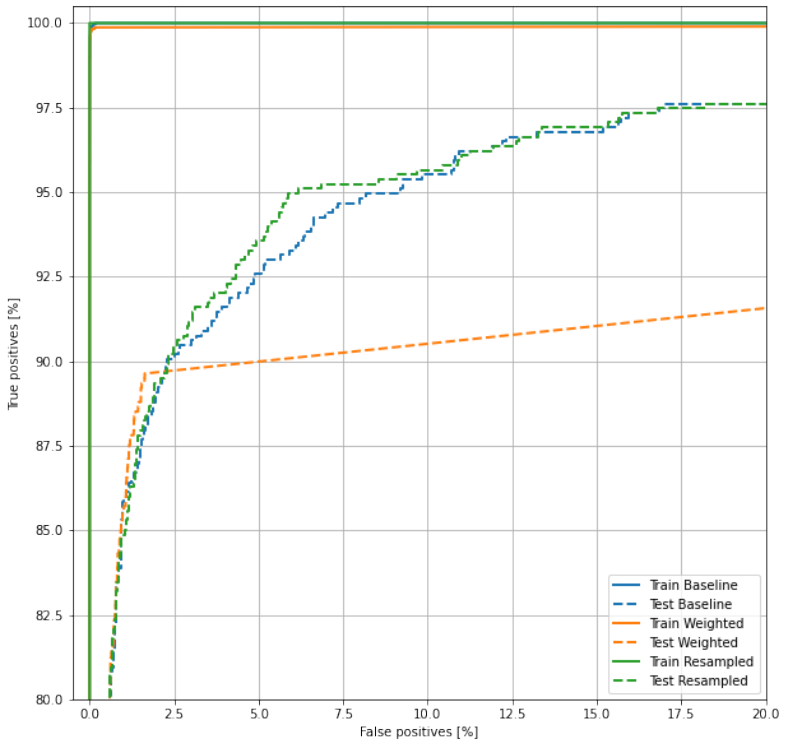

Figure 5.9: ROC of the model with 10 Dense layers and 9 Dropouts.

describes the results with the different models.

Using a model with 10 dense layers and 9 dropouts, and oversampled data achieved the best result so far, achieving an F1-score of 84.014% in the class *fear*. The only problem is the training time with the oversampled model that takes 18 minutes to train, but in this situation, if a quicker model is wanted the weighted model is the better one taking only 2 minutes and also obtaining a high F1-score result.

Analyzing Figure 5.9, it can be shown that all models, in the train, have an AUC of approximately 1. This means that the model can distinguish what is *fear* and what is not, so there is an slightly overfitting. The best model that achieves the best F1-score obtained an AUC equal to 0.9132 in the training and the weighted model appears to have a smaller AUC value in the training, relative to the other models.

```
Layer (type)            Output Shape              Param #
=================================================================
LSTM 1                  (None, 256, 256)          5758976
_________________________________________________________________
Dropout                 (None, 256, 256)          0
_________________________________________________________________
Dense                   (None, 256, 32)           8224
_________________________________________________________________
LSTM 2                  (None, 256)               295936
=================================================================
Total params: 6,063,136
```

Figure 5.10: RNN model.

5.3 Recurrent Neural Network

Since the Recurrent Neural Network has proved effective in time-based predictions, RNNs have been used to introduce another method. This approach has been embraced because films involve time, and fear is defined in more than one second.

A simple RNN with 2 LSTM's, a dense layer and a dropout was used. Figure 5.10 illustrates this RNN. The data was divided to form a 3D array, because the input to every LSTM layer must be three-dimensional.

Each dimensional of the 3D array consist of the samples, the time steps and the features [8]. In this test, the samples are the number of data that are possible to divide, for example, the number of samples are five if a movie have 1280 seconds and the number of seconds to feed the RNN are 256. The number of time steps is 256 seconds, and the features are the visual features sets.

The visual features sets were used to feed the model but the 2D data have to be reshaped and a time step of 256 has been used to do this. This 256 means that the model will be fed with 256 consecutive seconds in each sample. In most cases, the length of the movies is not a multiple of 256, and in these cases, the remaining seconds have been dropped. For example, if the movie has 1480 seconds, the last 200 seconds were dropped (256 seconds * 5 samples=1280 seconds and 1280 seconds + 200 seconds = 1480 seconds). Using this time step, the final 3D array results in the form of 309 samples and 5367 visual features (309, 256, 5367).

Using a Neural Network that requires a temporal sequence seems to be a good idea, but the outcomes have not been the best. The best result was a 9% F1-score in the *fear* class. As a result, it is concluded again that the database is not the best and does not have enough data to predict *fear*.

5.4 Summary

Deeper learning achieved even greater results compared to Chapter 4. These results were achieved using the random division of data, resulting in 84.014% of F1-score or IoU of 0.724 for fear prediction. Also, the data used was a combination of the visual and audio features sets. Compared with the Related Work (3.1) the results were much better because a different division of data was performed. Again, this leads to the conclusion that the *LIRIS-ACCEDE* database may not be fully representative of the universe of films.

Both RNN results were weak, in the case of the related work an IoU of 0.11992 was achieved and in the experiments carried out an F1-score of 9% was obtained. The features used were only the visual ones, and the methodology followed in [32] deployed a single-layer LSTM, while the experiments performed utilized 2 LSTM's.

Chapter 6

Conclusions and Future Work

The main goal of this thesis was to predict the induction of fear sensations triggered by audiovisual content, using machine learning techniques. The experiments reported here use the *LIRIS-ACCEDE* database, which was previously used in the *MediaEval* benchmarking initiative, in the scope of the *2018 Emotional Impact of Movies Task* challenge. Seven teams competed in this challenge producing outcomes for valence and arousal estimates, and for the forecast of *fear* inducing movie content, sharing similar goals as the ones of this dissertation. The *LIRIS-ACCEDE* database contains several movies, where each segment of one second was annotated as *fear* inducing segment or not. The work of this dissertation started by analyzing the approaches used by each one of these teams. Afterward, the first main task was to reorganize the data from *LIRIS-ACCEDE* in order to improve the efficiency of loading the data and prepare it for each experiment, because the data is divided in a second of film per file. At the end of this process, a Comma Separated Values (CSV) file for each movie set of features was created (for instance, the data for the first movie consists of 12 CSV's files representing the audio plus each visual feature set). This implementation was essential in order to optimize the data reading process required for feeding the machine learning classifiers.

The initial experiments were performed using the *Scikit-learn* library, which provided efficient implementations for most of the classical ML classification methods. Afterward, the *TensorFlow* library was used to create deeper neural network models, aiming at surpassing the performance of the previously applied ML methods.

The large size of the database was an important factor to take into consideration, not only because of memory issues but also because of execution time issues, since some of the experiments demanded long periods of training time. The models based on LSTMs demanded a significant memory size, as well as long periods of training, which turned out to be a problem considering the existing resources. As a result, the results achieved using such models were not optimized and, despite their potential, they turned out not to be the best performing ones.

It has been concluded that the *LIRIS-ACCEDE* dataset, despite being very large, does not contain a significant and representative set of movies for predicting emotions, such as fear. This conclusion is drawn because when having some information about the videos in the training and testing models the results are much better (the best was 84.0%) than those achieved when the models are trained and tested with disjoint film sets. Of all the experiments carried out, the best result was obtained by combining all the features provided, audio and visual, and trained with a model created with *TensorFlow*, containing 10 dense layers and an extensive application of dropout. Through this model, an answer to the question presented in the introduction is reached, concluding that deep learning is more fitting for this type of problem because by increasing the size of the Neural Network from 4 to 10 dense layers, better results are achieved. It is also worth noting that visual feature sets have achieved a better outcome compared to audio, and Auto Color Correlation (acc) is the best visual feature set. When the model is trained with all the features combined, the f1-score for fear prediction, goes from 64% (all visual features) to 72.5%, combining the audio, results in an 8.5% improvement.

Concerning future directions, efforts should be made for producing larger databases of annotated movies, in order to achieve a more significant content variability. With the current dataset, we can train a model with samples associated to film segments and to correctly predict fear induction in different segments of the same film. However, training with a selection of movies and generalizing predictions for a different selection did not achieve satisfactory results. Another topic that deserves further investigation is the feature extraction process. Current visual features are based on key-frames, and therefore they do not consider temporal movie characteristics, which may also be relevant to trigger emotional states. Additionally, the use of Recurrent Neural Networks should be revisited as more data becomes available.

Bibliography

[1] Soltius (https://stats.stackexchange.com/users/201218/soltius). *How is it possible that validation loss is increasing while validation accuracy is increasing as well*. Cross Validated. URL:https://stats.stackexchange.com/q/341054 (version: 2020-12-17). eprint: https://stats.stackexchange.com/q/341054. URL: https://stats.stackexchange.com/q/341054 (visited on 12/29/2020).

[2] Charu C. Aggarwal. "Educational and software resources for data classification". In: *Data Classification: Algorithms and Applications* (2014), pp. 657–665. DOI: 10.1201/b17320.

[3] AudEERING. *OpenSMILE - audEERING*. 2019. URL: https://www.audeering.com/opensmile/ (visited on 10/07/2020).

[4] Neelima Bagri and Punit Johari. "A Comparative Study on Feature Extraction using Texture and Shape for Content Based Image Retrieval". In: *International Journal of Advanced Science and Technology* 80 (July 2015), pp. 41–52. DOI: 10.14257/ijast.2015.80.04.

[5] Isaac N Bankman, Thomas S Spisz, and Sotiris Pavlopoulos. "Chapter 15 - Two-Dimensional Shape and Texture Quantification". In: ed. by ISAAC N B T - Handbook of Medical Image Processing BANKMAN and Analysis (Second Edition). Burlington: Academic Press, 2009, pp. 261–277. ISBN: 978-0-12-373904-9. DOI: https://doi.org/10.1016/B978-012373904-9.50024-6. URL: http://www.sciencedirect.com/science/article/pii/B9780123739049500246 (visited on 10/27/2020).

[6] Elissavet Batziou et al. "Visual and audio analysis of movies video for emotion detection @ Emotional Impact of Movies task MediaEval 2018". In: *CEUR Workshop Proceedings* 2283.October (2018), pp. 29–31. ISSN: 16130073.

[7] Jason Brownlee. *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning*. 2017. URL: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/ (visited on 09/19/2020).

[8] Jason Brownlee. *How to Reshape Input Data for Long Short-Term Memory Networks in Keras*. 2017. URL: https://machinelearningmastery.com/reshape-input-data-long-short-term-memory-networks-keras/ (visited on 10/25/2020).

[9] Amar Budhiraja. *Dropout in (Deep) Machine learning*. URL: https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5 (visited on 10/30/2020).

[10] Bugraakyildiz. *Industry Similarity via Jaccard Index*. 2015. URL: https://axialcorps.wordpress.com/2015/05/01/industry-similarity-via-jaccard-index/ (visited on 01/07/2020).

[11] Savvas Chatzichristofis and Yiannis Boutalis. *CEDD: Color and Edge Directivity Descriptor: A Compact Descriptor for Image Indexing and Retrieval*. 2008, pp. 312–322.

[12] Savvas Chatzichristofis and Yiannis Boutalis. "FCTH: Fuzzy Color and Texture Histogram - A Low Level Feature for Accurate Image Retrieval". In: *WIAMIS 2008 - Proceedings of the 9th International Workshop on Image Analysis for Multimedia Interactive Services*. 2008, pp. 191–196. ISBN: 978-0-7695-3344-5. DOI: 10.1109/WIAMIS.2008.24.

[13] Savvas A. Chatzichristofis, Avi Arampatzis, and Yiannis S. Boutalis. "Investigating the behavior of compact composite descriptors in early fusion, late fusion and distributed image retrieval". In: *Radioengineering* 19.4 (2010), pp. 725–733. ISSN: 12102512.

[14] L Universite Claude and Bernard Lyon. "PHD THESIS D ´ etection des ´ par Rizwan Ahmed KHAN". In: (2013).

[15] Yann N. Dauphin, Harm De Vries, and Yoshua Bengio. "Equilibrated adaptive learning rates for non-convex optimization". In: *Advances in Neural Information Processing Systems* 2015-Janua.April (2015), pp. 1504–1512. ISSN: 10495258. arXiv: 1502.04390.

[16] Emmanuel Dellandréa et al. "The MediaEval 2018 emotional impact of Movies task". In: *CEUR Workshop Proceedings* 2283 (2018), pp. 6–8. ISSN: 16130073.

[17] Jonas DeMuro. "What is a neural network?" In: IOP Publishing Ltd, 2019. DOI: 10.1887/0750303123/b365c4. URL: https://www.techradar.com/news/what-is-a-neural-network (visited on 01/06/2020).

[18] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[19] Google Developers. *Classification: ROC Curve and AUC*. 2019. URL: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc (visited on 08/16/2020).

[20] Cheng-Jin Du and Da-Wen Sun. *Computer Vision Technology for Food Quality Evaluation*. 2008. URL: https://www.sciencedirect.com/topics/computer-science/decision-tree-classifier (visited on 10/16/2020).

[21] Sydney Firmin. *Tidying up with PCA: An Introduction to Principal Components Analysis*. 2019. URL: https://towardsdatascience.com/tidying-up-with-pca-an-introduction-to-principal-components-analysis-f876599af383 (visited on 01/09/2020).

[22] R Gandhi. *Support Vector Machine — Introduction to Machine Learning Algorithms*. 2018. URL: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 (visited on 01/08/2020).

[23] Onel Harrison. *Machine Learning Basics with the K-Nearest Neighbors Algorithm*. URL: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761 (visited on 10/23/2020).

[24] Jie Hu et al. "Squeeze-and-Excitation Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.8 (2020), pp. 2011–2023. ISSN: 19393539. DOI: 10.1109/TPAMI.2019.2913372. arXiv: 1709.01507.

[25] ImageNet. *About ImageNet*. URL: http://image-net.org/about-overview (visited on 11/06/2020).

[26] Muhammad Imran et al. "Class wise image retrieval through scalable color descriptor and edge histogram descriptor". In: (2016). URL: http://www.science-gate.com/IJAAS/V3I12/Imran.html (visited on 10/15/2020).

[27] Chayan Kathuria. *Regression — Why Mean Square Error?* 2019. URL: https://towardsdatascience.com/https-medium-com-chayankathuria-regression-why-mean-square-error-a8cad2a1c96f (visited on 01/08/2020).

[28] Keras. *ResNet-50*. 2015. URL: https://www.kaggle.com/keras/resnet50 (visited on 10/18/2020).

[29] Aditya Khosla. *Places2: A Large-Scale Database for Scene Understanding*. 2016. URL: http://places2.csail.mit.edu/download.html (visited on 10/07/2020).

[30] Tobey H. Ko et al. "Towards learning emotional subspace". In: *CEUR Workshop Proceedings* 2283.3 (2018), pp. 4–6. ISSN: 16130073.

[31] Miroslav Kubat. "An Introduction to Machine Learning". In: *An Introduction to Machine Learning* (2017), pp. 1–348. DOI: 10.1007/978-3-319-63913-0.

[32] Chloe Loughridge and Julia Moseyko. "IM-JAIC at MediaEval 2018 Emotional Impact of Movies Task Chloe". In: *CEUR Workshop Proceedings* 2283.October (2018). ISSN: 16130073.

[33] A Agnes Lydia and F Sagayaraj Francis. "Adagrad: An Optimizer for Stochastic Gradient Descent". In: *International Journal of Information and Computer Science* 6.5 (2019), pp. 566–568. arXiv: 1609.04747. URL: http://ijics.com.

[34] Ec-lyon.fr. *LIRIS-ACCEDE*. 2015. URL: https://liris-accede.ec-lyon.fr/database.php (visited on 01/07/2020).

[35] Ye Ma, Xihao Liang, and Mingxing Xu. "THUHCSI in MediaEval 2018 Emotional Impact of Movies Task". In: *CEUR Workshop Proceedings* 2283 (2018). ISSN: 16130073.

[36] J Magiya. *Pearson Coefficient of Correlation Explained*. 2019. URL: https://towardsdatascience.com/pearson-coefficient-of-correlation-explained-369991d93404 (visited on 01/07/2020).

[37] Pedro Marcelino. *Transfer learning from pre-trained models*. 2018. URL: https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751 (visited on 09/05/2020).

[38] Martin Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/ (visited on 09/26/2020).

[39] Mathworks.com. *Understanding Support Vector Machine Regression*. 2019. URL: https://www.mathworks.com/help/stats/understanding-support-vector-machine-regression.html (visited on 01/15/2020).

[40] Mathworks.com. *vgg16*. URL: https://www.mathworks.com/help/deeplearning/ref/vgg16.html (visited on 09/01/2020).

[41] Mathworks.com. *What Is Deep Learning? 3 things you need to know*. 2019. URL: https://www.mathworks.com/discovery/deep-learning.html (visited on 07/02/2020).

[42] Aditi Mittal. *Understanding RNN and LSTM*. 2019. URL: https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e (visited on 10/15/2020).

[43] Multimediaeval.org. *About MediaEval*. 2019. URL: http://www.multimediaeval.org/about/ (visited on 11/06/2019).

[44] Sarang Narkhede. *Understanding AUC - ROC Curve*. URL: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5 (visited on 10/29/2020).

[45] Naxingyu. *naxingyu/opensmile*. 2019. URL: https://github.com/naxingyu/opensmile (visited on 10/11/2020).

[46] Neurohive.io. *VGG16 - Convolutional Network for Classification and Detection*. 2018. URL: https://neurohive.io/en/popular-networks/vgg16/ (visited on 09/07/2020).

[47] Aleksander Obuchowski. *Understanding neural networks 1: The concept of neurons*. 2019. URL: https://becominghuman.ai/understanding-neural-networks-1-the-concept-of-neurons-287be36d40f (visited on 07/10/2020).

[48] Dong Kwon Park, Yoon Seok Jeon, and Chee Sun Won. "Efficient use of local edge histogram descriptor". In: (2000), pp. 51–54. DOI: 10.1145/357744.357758.

[49] Ken Peffers et al. "A design science research methodology for information systems research". In: *Journal of Management Information Systems* 24.3 (2007), pp. 45–77. ISSN: 07421222. DOI: 10.2753/MIS0742-1222240302.

[50] Places2.csail.mit.edu. *Places: A 10 million Image Database for Scene Recognition*. URL: http://places2.csail.mit.edu/ (visited on 10/18/2020).

[51] Wichian Premchaiswadi and Anucha Tungkasthan. *A compact auto color correlation using binary coding stream for image retrieval*. 2011, pp. 430–436.

[52] Khanh An C. Quan, Vinh Tiep Nguyen, and Minh Triet Tran. "Frame-based evaluation with deep features to predict emotional impact of movies". In: *CEUR Workshop Proceedings* 2283 (2018), pp. 2–4. ISSN: 16130073.

[53] Quora.com. *What is the deep neural network known as "ResNet-50"? - Quora*. 2014. URL: https://www.quora.com/What-is-the-deep-neural-network-known-as-%E2%80%9CResNet-50%E2%80%9D (visited on 11/06/2020).

[54] Sebastian Raschka. *Linear Discriminant Analysis*. 2014. URL: https://sebastianraschka.com/Articles/2014%7B%5C_%7Dpython%7B%5C_%7Dlda.html (visited on 09/09/2020).

[55] A. Rosebrock. *Intersection over Union (IoU) for object detection*. 2016. URL: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/ (visited on 01/06/2020).

[56] Tim Ruscica. *Introduction to Neural Networks*. 2020. URL: https://colab.research.google.com/drive/1m2cg3D1x3j5vrFc-Cu0gMvc48gWyCOuG%7B%5C#%7DforceEdit=true%7B%5C&%7DsandboxMode=true%7B%5C&%7DscrollTo=jqVqT%7B%5C_%7DCxh4Ho (visited on 06/08/2020).

[57] Salouan S. Safi and B. Bouikhalene. "Printed Noisy Greek Characters Recognition Using Hidden Markov Model, Kohonen Network, K Nearest Neighbours and Fuzzy Logic". In: *International Journal of Signal Processing, Image Processing and Pattern Recognition* 8.10 (2015), pp. 241–256. ISSN: 20054254. DOI: 10.14257/ijsip.2015.8.10.26.

[58] Sumit Saha. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. 2018. URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (visited on 01/05/2020).

[59] SBCoaching. *Emoções: definição, tipos e importância de se ter o controle*. 2019. URL: https://www.sbcoaching.com.br/blog/emocoes/ (visited on 01/05/2020).

[60] Scribd. *Gradient Descent - Problem of Hiking Down a Mountain: Derivatives*. URL: https://pt.scribd.com/document/456344322/Gradient-Descent (visited on 09/20/2020).

[61] Sweta Shaw. *RMSprop : A Better Way to Optimize Your Model.* 2019. URL: https://medium.com/@shwetaka1988/rmsprop-a-better-way-to-optimize-your-model-bc4eaca33090 (visited on 10/05/2020).

[62] Koo Ping Shung. *Accuracy, Precision, Recall or F1?* 2018. URL: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9 (visited on 07/19/2020).

[63] Lindsay Smith. "A tutorial on PCSA". In: *Department of Computer Science, University of Otago.* (2006), pp. 12–28. DOI: http://www.cs.otago.ac.nz/research/techreports.php.

[64] Saishruthi Swaminathan. "Logistic Regression — Detailed Overview". In: (). URL: https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc (visited on 10/20/2020).

[65] Expert System. *What is Machine Learning? A definition*. 2017. URL: https://expertsystem.com/machine-learning-definition/ (visited on 10/02/2020).

[66] Christian Szegedy et al. "Inception v4". In: *Population Health Management* 18.3 (2015), pp. 186–191. ISSN: 1942-7891. arXiv: 1409.4842v1. URL: http://online.liebertpub.com/doi/10.1089/pop.2014.0089.

[67] TensorFlow. *Recurrent Neural Networks (RNN) with Keras*. 2020. URL: https://www.tensorflow.org/guide/keras/rnn (visited on 09/20/2020).

[68] Statistics How To. *Mean Squared Error: Definition and Example - Statistics How To*. 2013. URL: https://www.statisticshowto.com/mean-squared-error/ (visited on 10/20/2020).

[69] www.oreilly.com. *Mean squared error - Hands-On Machine Learning for Cybersecurity [Book]*. URL: https://www.oreilly.com/library/view/hands-on-machine-learning/9781788992282/3da539f8-3925-47d7-b3dd-61ae8420c8e8.xhtml (visited on 10/20/2020).

[70] Www.sciencedirect.com. "Local Binary Pattern". In: (). URL: https://www.sciencedirect.com/topics/engineering/local-binary-pattern (visited on 10/15/2020).

[71] www.semanticscholar.org. *Color layout descriptor*. URL: https://www.semanticscholar.org/topic/Color-layout-descriptor/36875 (visited on 10/15/2020).

[72] Rikiya Yamashita et al. "Convolutional neural networks: an overview and application in radiology". In: *Insights into Imaging* 9.4 (2018), pp. 611–629. ISSN: 18694101. DOI: 10.1007/s13244-018-0639-9.

[73] Yun Yi, Hanli Wang, and Qinyu Li. "CNN features for emotional impact of movies task". In: *CEUR Workshop Proceedings* 2283.October (2018), pp. 29–31. ISSN: 16130073.

[74] Tony Yiu. *Understanding neural networks*. 2019. DOI: 10.2307/1270439. URL: https://towardsdatascience.com/understanding-neural-networks-19020b758230 (visited on 01/09/2020).

[75] Liang Chih Yu et al. "Building Chinese affective resources in valence-arousal dimensions". In: *2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016 - Proceedings of the Conference* June (2016), pp. 540–545. DOI: 10.18653/v1/n16-1066.

[76] Xiaotong Zhang et al. "Imbalance learning-based framework for fear recognition in the mediaeval emotional impact of movies task". In: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH* 2018-September.September (2018), pp. 3678–3682. ISSN: 19909772. DOI: 10.21437/Interspeech.2018-1744.

[77] Jaime Zornoza. *Logistic Regression Explained*. 2020. URL: https://towardsdatascience.com/logistic-regression-explained-9ee73cede081 (visited on 10/18/2020).

Noname manuscript No.
(will be inserted by the editor)

Automatic detection of fear-inducing segments in videos

André Maia · Tomás Brandão · Fernando Batista

06 January 2021

Abstract This paper explores the use of machine learning to predict the emotional effects triggered by video, more specifically, the automated recognition of fear-inducing segments. *LIRIS-ACCEDE* dataset is used in the experiments because it includes several sets of features annotated as fear, which are extracted for every second of each movie. Both classical and deep learning techniques have been implemented and evaluated, using the *scikit-learn* and *TensorFlow* machine learning libraries.

Two different approaches for training and testing have been followed: film-level dataset splitting, where different films were used for training and testing; and sample-level dataset splitting, which allowed that different samples coming from the same films were used for training and testing. The experimental results indicate that *LIRIS-ACCEDE* dataset does not contain a significant and representative set of movies for predicting emotions, such as fear because from the two approaches performed of splitting the data the second approach achieved a F1-score of about 84.0% instead of 18.5% from the first approach.

Keywords Machine Learning & Emotional Prediction, Fear Prediction, Video Classification, Deep Learning

1 Introduction

Human beings, according to scientists, can manifest up to twenty seven distinct emotions SBCoaching 2019, and human emotions can be mapped into six different categories: Happy; Sad; Angry; Fear/Worry; Surprise; and Disgust, proposed by Ekman (Claude and Lyon, 2013). Machine learning (ML) is a technique used to handle difficult tasks, such as detecting the emotional impact of movies, face recognition, data mining, among others, and can solve very complicated tasks taking much less time than a human Kubat 2017. Automatically detecting fear induced by a film is a difficult task

andre_maia@iscte-iul.pt

tomas.brandao@iscte-iul.pt

fernando.batista@iscte-iul.pt

and ML can help to perform this detection, which can be very beneficial having many advantages, such as:

- Video summarization mechanisms that take emotional factor into consideration, summarizing the video according to excerpts that represent emotional states.
- Improving the recommendation mechanisms and personalization of video transmission services. For instance, to evaluate a film in order to produce tags according to the triggered emotional states.
- Protect viewers from content that could affect them, such as children and more vulnerable people. For example, filter material to protect a child who surfs the Internet watching a variety of videos, by blocking content that may include violent or disturbing scenes.

When watching movies, feelings and states of mind are induced into the viewers. Since no one is identical, the induced emotions may differ from person to person. Analyzing the emotional impact of a video clip on audiences may also be used to strengthen or monitor the psychological influence of media on persons, to maximize audience interaction with media material, or to create customized media content Quan, Nguyen, and Tran 2018.

The key purpose of this article is to determine the emotional effect of audiovisual content on individuals, more specifically the fear. A dataset named *LIRIS-ACCEDE* Ec-lyon.fr 2015, which provides a total of 56 movies, corresponding to approximately to 24 hours, with annotations focused on fear, valence and arousal, was used. This database already includes several sets of features, extracted for every second of each movie, such as color and texture visual features, audio features, etc., providing a total of twelve features sets containing a large amount of features.

Different approaches are proposed in order to evaluate the results, which feature sets and ML models lead to the best results in predicting the fear inducing segments of the movies. Two separate training and testing methods are used: the first one is to split the data at film level, getting samples coming from different films for training and testing, while the second is less restrictive, randomly splitting the data at the sample level, causing samples of all movies to appear in the train and test sets.

Two Python libraries, called *Scikit-learn* and *TensorFlow*, are used for the estimation of fear inducing film segments. Seven *scikit-learn* classifiers are used for the implementation of classical ML models, such as Gaussian Naive Bayes, Decision Tree Classifier, Logistic Regression, among others; *TensorFlow* is used for the development of deep neural networks.

2 Related work

The starting point for this article were the ideas behind a "benchmarking initiative dedicated to evaluating new algorithms for multimedia access and retrieval" called *MediaEval* (Multimediaeval.org, 2019). *MediaEval* organizes several challenges on image and video recognition tasks. Among them, the one called *The MediaEval 2018 Emotional Impact of Movies Task* is very close to the purpose of this article, and therefore it will be further analyzed and studied.

The main objective *MediaEval 2018 Emotional Impact of Movies Task* is to create a machine learning based program that automatically predicts the emotional impact that video content causes in its viewers (Dellandréa et al., 2018). The task's participants are

supposed to deploy multimedia features and models that allow to predict the emotion felt by most of the audience watching the movie. Each participant team should consider two scenarios as sub-tasks:

1. Valence and Arousal prediction: it's expected to continuously predict a score of valence and arousal (*i.e.*, every second) along the duration of the movie subject to analysis. Valence is classified as a scale, form the most unpleasant to the most pleasant emotional state, while Arousal is quantified from the calmest to the most exciting emotional states;
2. Fear detection: the participants are required to present a method that allows to predict the beginning and ending times of fear inducing sequences within the movies.

Different teams proposed different methods to handle the challenge. The sub-task mentioned in the following lines is fear detection, which defines methods that classify each sample individually from the remaining ones, and approaches that take advantage of potential temporal dependencies of the samples, classifying sample sequences and not isolated ones. All approaches performed 4 Runs which will be described individually.

The approach proposed in (Yi, Wang, and Li, 2018) uses four sets of features that could influence emotions: audio, action, object and scene features, which are extracted using pre-trained convolutional neural networks. They performed 4 runs and each one is consisted of:

1. Features provided by *Mediaeval*
2. Audio and scene features
3. Audio, scene and object features
4. Audio, scene and action features
5. Audio, scene, object and action features

The audio features were extracted using VGGish (Martin Abadi et al., 2015). A CNN is used to extract the action features, which contains two separate recognition streams, spatial and temporal. Object features are obtained by a CNN called the Squeeze-and-Excitation Network (SENet). Lastly, pre-trained ResNet-50 model on Places365 extract the scene features.

The best result was obtained by Run 4, using audio, scene and action features, achieving IoU value of 0.1575.

Another strategy was proposed in (Batziou et al., 2018), followed by 4 Runs for the fear subtask:

1. Run 1 consists of visual based classification and uses transfer learning approach based on the pre-trained VGG16 model for Places2 dataset;
2. Run 2 is similar to Run 1, with additional post-processing to remove isolated predictions;
3. Run 3 is based on audio features only, using a neural network with 3 hidden layers for classification;
4. Run 4 also uses a NN combining the audio features with the visual features provided in *LIRIS-ACCEDE* (subject to PCA).

The best result was obtained by Run 1 achieving a value of IoU equal to 0.075.

In the work proposed by Ko *et al.* in (Ko et al., 2018) Bias Discriminatory Embedding algorithm (BDE) was used to learn fear. Each Run is composed by a D-dimensional target subspace with $D = 4, 5, 9, 10$, respectively. The outcome was quite unsatisfactory obtaining 0.1052 of IoU, which may be due to a high dataset imbalance between the *fear* and *no-fear* classes.

The last work performed by (Ma, Liang, and Xu, 2018), implement time-sequential models, and each run uses LSTMs. The best result achieved 0.11992 of IoU, defined as a single-layer LSTM with visual features.

3 Classical Classifiers and Feature Analysis

This chapter describes the experiments carried out to evaluate which sets of features and classifiers lead to the best results in fear detection. The details of the experimental setup can be found in Section 3.1.

In the first round of experiments, reported in Section 3.2, training and test sets are disjoint at video level, meaning that the samples in the training set come from different videos than those used in the testing set. However, this training/testing methodology lead to poor classification performance, suggesting that the the videos used in the training set are not representative of those used in the test set.

In order to tackle this problem, another approach was followed by using samples from the same videos for training and for testing. *i.e.*, training and test sets are disjoint at sample level, and thus with less restrictions than the first approach. Both the training an test sets will therefore contain samples taken from the same videos, meaning that, on the same video, a set of samples is randomly assigned to the training set while the remaining ones are assigned to the test set. The experiments that follow this approach are described in Section 3.3.

3.1 Experimental Setup

This section describes the experimental setup performed to identify feature sets that are most relevant for the fear prediction. The *LIRIS-ACCEDE* is the main dataset used in this article and consists of a collection of 160 films made by professionals and amateurs, shared under Creative Commons licenses that allow content redistribution. From the full film set, 56 which contains annotations regarding the fear-inducing movie parts, as well as values for valence and arousal are the ones used. Audio features have been extracted with the Smile toolbox, considering a 5-second window sliding through the entire movie, with one-second shifts, leading to one audio feature vector per video second, resulting in 1583 features. Video features, were computed from key frames extracted from every second of video. In total, it includes eleven sets of visual features and a single of audio feature.

As for the machine learning classifiers used in the experiments reported in this chapter, the following have been used:

- Linear Discriminant Analysis.
- Decision Tree Classifier.
- K-Nearest Neighbor.
- Multi-layer Perceptron (MLP) Classifier (Neural Network).
- Gaussian Naive Bayes Classifier.
- Logistic Regression Classifier.
- Linear Support Vector Classifier.

All these classifiers were tested in order to evaluate which ones lead to the best results in the fear inducing prediction task. Their implementations are available in the *scikit-learn* python library.

The other main goal of this chapter is to determine which feature sets, or feature set combinations, are the most suitable for the task of predicting fear inducing movie segments. To accomplish this task, the following flow of experiments was performed:

1. Each individual set of features available in *LIRIS-ACCEDE* was, in turn, used as the classifiers input. The goal was to evaluate the discriminating power of each feature set.
2. The feature sets leading to the best results in the previous step were combined and used as the classifiers input.
3. All visual features sets were combined into a single feature vector used as input for all tested classifiers.
4. Identical to the previous step, but in this case the training set was additionally subject to a class balancing procedure .
5. Finally, all audio and visual features sets were combined.

All the experiments share the same principle: to train and predict the presence of fear-inducing video samples using the seven machine learning classifiers and variations in the feature data used as the classifiers input.

3.2 Using Different Films for Training and Testing

This section reports the experiments done using a training and test sets are composed of completely different sets of videos. The principle is that the video materials were initially split into 44 videos for training and the remaining 12 for testing. The training set contains a total of 55251 samples (one sample for each second of video), covering approximately 15 hours of video; as for the testing set, it contains 32205 samples, covering approximately 9 hours of video. The experiments described in Section 3.1 were then performed.

Since the number of samples with fear is much smaller than the number of samples without fear, accuracy is not a good evaluation metric. In conclusion, since the dataset classes are unbalanced the F1-score was used as the assessment metric, as it relies on the precision and recall metrics to compute the final outcome. The IoU metric is also robust to class unbalancing and therefore it is also an adequate performance assessment metric for the *fear* prediction task.

The best results from the experiments described in Section 3.1 are represented in Table 1. The F1-score column depicts two values separated by '/': the first value reflects the F1-score for predicting the *fear* inducing class, while the second value is the F1-score for predicting the *no-fear* class.

The first experiments consisted of training a set of *scikit-learn* classification algorithms, synthesized at the beginning of this chapter, using features coming from individual sets of visual and audio features. The goal was to find out which of the feature sets are the most promising ones, when the task is to predict the temporal video intervals that induce fear into the viewer. Then best feature sets, which are *cl*, *eh*, *fc6* and *sc* (achieving 14.3, 14.3, 13.5 and 18.5 per cent of *fear* prediction, respectively), were combined in order to check if better results are obtained by using a larger amount of features for classification. These feature sets represent the color of the image, the edge types in an image, values extracted using VGG16 CNN, and the color signatures, respectively. Class balancing with all visual features sets was done because the 87456

Method	Algorithm	F1-score	IoU	Time to train and test (s)
Individual features - *sc*	GaussianNB	0.185/ 0.928	0.1017	0,095
All features combined		0.152/ 0.679	0.0820	12,717
Individual features - *cl*		0.143/ 0.828	0.0771	0,054
Individual features - *eh*		0.143/ 0.759	0.0772	0,122
Best features combined		0.137/ 0.624	0.0734	8,750
Class balancing	KNN	0.133/ 0.763	0.0710	1017,139
	GaussianNB	0.127/ 0.549	0.0680	4,220
Visual feature sets combined		0.126/ 0.546	0.0670	10,612

Table 1 Best results.

		Predicted label	
		no-fear	*fear*
Actual label	*no-fear*	27438	3161
	fear	1121	485

Table 2 GaussianNB confusion matrix of *sc* feature set.

samples in the dataset, 83759 samples were labeled as *no-fear* (95.8%), and 3697 samples as *fear* (4.2%). This fact may contribute to the reason why several experiments produced a very small amount of classifications that fall in the *fear* class.

Table 1 synthesizes the performance achieved for the most promising eight setups, from best to the worst. All these methods obtained the best result by using a Gaussian Naive Bayes classifier except one.

Inspecting Table 1, the global outcome is poor, at 18.5% of F1-score by predicting the label *fear*. The time required to train and test are very low, but in the end, the best method (Individual features - *sc*) obtained one of the shortest times. Furthermore, the use of a single feature set requires less time for training, and memory consumption is minimized. Since the results are getting worse as more features are used, it can be speculated that, as more feature sets are used, the more noticeable becomes the overfitting to the training set. Remember that samples used for training come from different films than those used for testing, which suggests that the training set is not representative of the film universe. It is noteworthy that the KNN algorithm has obtained the longest time to test and train, compared with the others. Some algorithms achieved a poorer performance, in some cases the F1-score to predict *fear* was 0%.

The experiments suggest dataset limitations. The dataset classes are very unbalanced, which causes that most models are prone to forecasting the *no-fear* class since this class represents about 95.8% of samples used in training. On the other hand, balancing the data by removing elements of the most numerous class lead to a rather small training set.

The experiments also showed that the use of a larger amount features did not translate into better performance. This fact may be due to overfitting problems that become more evident as more features are used.

Analyzing Table 2, which represents the confusion matrix of the Gaussian Naive Bayes algorithm and the feature set *sc*, it was able to correctly predict 485 out of 1606 *fear* samples, and 3161 *no-fear* samples were mispredicted as *fear*.

Method	Algorithm	F1-score	IoU	Time to train and test (min)
All features combined	MLP	0.725/ 0.988	0.569	51.36
Best features combined	KNN	0.650/ 0.987	0.481	172.95
All visual features combined	MLP	0.640/ 0.985	0.471	307.54
Individual features - acc	KNN	0.637/ 0.986	0.467	6.75
All visual features combined		0.631/ 0.986	0.461	105.80
Individual features - fc6		0.619/ 0.986	0.034	217.91
Individual features - jcd		0.599/ 0.985	0.024	5.43
Class balancing	MLP	0.316/ 0.910	0.188	1.83

Table 3 Best results.

3.3 Using Samples From Same Films for Training and Testing

The experiments done in Section 3.2 showed that the data used is cleary insufficient which makes the models do not generalize well. In order to overcome this problem, this section describes a new set of experiments at the sample level. A portion of the movie samples are used for training, and another for testing, and this is done by randomly split the data. The algorithm's training and test processes will therefore access samples coming from the full films dataset.

The main database containing a total of 87456 samples was randomly split into 20% of all samples for testing (17492 samples) and 80% for training (69964 samples).

The flow of experiments previously performed in Section 3.2 was also performed for this new dataset division, in order to find out which algorithms and feature sets lead to the best results. The following subsections depict the experimental results.

By observing Table 3, it can be seen that the results obtained are much better than those depicted in Section 3.2. With the new training/testing set division, the best F1-score regarding the prediction of *fear* is much higher. Training and testing with samples coming from the same films lead to F1-score of 72.5% predicting the *fear* tag and 99.8% predicting the *no-fear*, for the MLP Classifier, when all the available features (visual and audio) were combined.

When compared to the results presented in Section 3.2, the reason for this shift to higher F1-scores is due to the new training/ test set division. In this experiment, the training set contains samples coming from all the films available in the dataset, and therefore it is a richer set. On the other hand, the test set, although consisting of samples different from those used for training set, contains samples that are expected to be more correlated with those used for training, than in the situation that they were coming from different films. The only thing that got worse was the training and testing time, going from a few seconds to, in some cases hours.

The results were not the best with a class balancing, as the best result is 38% well below the previous ones, so a very poor result is obtained by this approach. This may have happened because important data was removed.

An inspection of the confusion matrix in Table 4 relative to all features combined and MLP algorithm, reveals that the *fear* class was well classified 538 times and badly 198, and the *no-fear* class was correctly predicted 16546 times.

		Predicted label	
		no-fear	*fear*
Actual label	*no-fear*	16546	198
	fear	210	538

Table 4 MLP confusion matrix of all features sets combined.

Fig. 1 Three frames from the movie 0.

Fig. 2 Second segment of the movie 0.

3.4 Summary

Some conclusions can be drawn from these experiments, because when comparing the use of training/test set division at film level with division at sample level (3.2 vs 3.3), the latter achieved better results. F1-scores improved from 0.185 to 0.725. When training and testing with distinct films there are far too many classification errors, meaning that learning based on a set of films assigned to the training set did not generalize for different films, and this is a very bad result which it is possible to say that the model can't predict *fear*. On the other hand, the predictions were much more accurate when the ML algorithms were trained with about 80% of the samples coming from all the videos present in the dataset, meaning that generalization is achieved for different moments of the films. These observations suggest that the *LIRIS-ACCEDE* database may not be sufficiently representative of the films universe. A larger dataset covering more film variations should help to improve the learning processes.

4 Fear-inducing Classifications

This section describes some classifications that induce fear and, for that, some of the films were watched manually to verify that the *fear* tag was placed correctly and the predictions made by the different models were analyzed.

A function was created to generate subtitles in order to be able to analyze the tag in the films. A subtitle that says "fear" has been applied to the videos.

Figure 1 shows three frames from the movie *MEDIAEVAL18 00* and reveal that the first two frames (explosion and body) convey fear, but are not identified as such, and the third frame, which contains a person's body, is correctly tagged as fear.

Fig. 3 Segment of the movie 16.

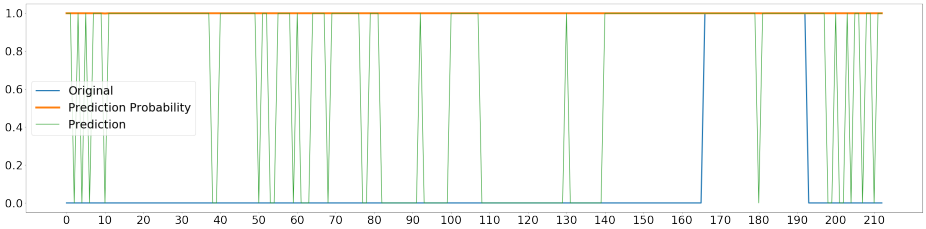

Fig. 4 Visual features predictions performed by GaussianNB.

Figure 2 shows an image that transmits fear, but the audio contains a piece of happy music, so the audio does not correspond with the current action, thus the audio, in this case, was misleading to recognize fear.

The *fear* tag in the dataset contains scenes like a fight and with suspense, images with blood, pistols, but there are some cases where this doesn't happen. This is illustrated in Figure 3, which shows a person placed in a bag with a man carrying a gun in his pocket, and this frame has no annotation of *fear*.

The conclusion is that the *fear* annotations are not consistent. In some cases, the *fear* tag is well-associated, and some are not, plus there are some cases of fear that do not exist. Feeling of fear can also vary person to person, so it's hard to tell for certain whether or not a picture involves fear.

Graphics were made for each film, in order to examine the predictions made by the best models, represented in the Section 3.2 and 3.3, GaussianNB and MLP. This graph represents, in each second, the real tag *fear* or *no-fear* (1 or 0 in the axis y), the prediction that the model performs (green line) and the confidence in each prediction (orange). This likelihood is the assurance that the model has, for example, if the model says that in the first second the probability to predict *fear* is 1, this means that the model has 100% confidence that fear occurs at that time. Figure 4 corresponds to movie 54 in the dataset of *LIRIS-ACCEDE*. The x axis represents each second of the movie, in this case the movie has 213 seconds, and these predictions were made using visual features and GaussianNB.

Analyzing Figure 4 is it possible to see that the model outputs a lot of random predictions at the beginning of the movie, the green line goes up and downs (classifies as *fear* and *no-fear*) at a time when there is no fear conveyed. In all of these predictions, the model has a 100% predictability, which is definitely not a positive indication. Looking at these results, it is possible to conclude that this is not a good model since it performs a lot of *fear* predictions with only one second. In one second exists fear, and the next one doesn't. This doesn't occur in movies, if the movies transmit fear, that fear will be transmitted in the next seconds and not in one unique second. In this

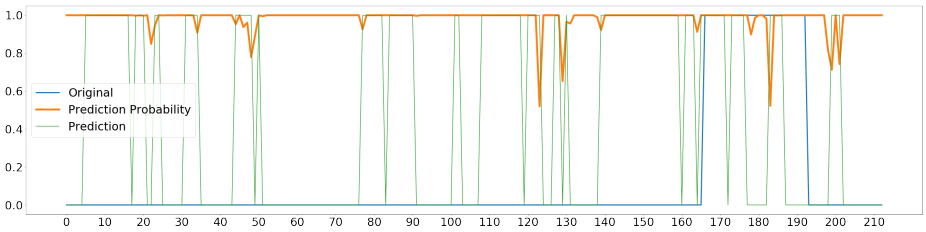

Fig. 5 Audio features predictions performed by GaussianNB.

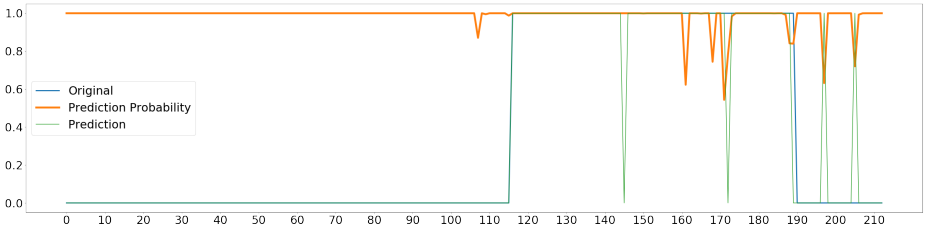

Fig. 6 Visual features predictions performed by MLP.

movie, it is possible to see that exists fear in the seconds 165 to the 193, and in all graph, there's no peak of fear in a unique second.

With the audio features, represented in Figure 5, it's possible to see the same thing, but with these features, the predictability isn't necessarily 100%.

Figure 6 represents the predictions performed with the visual feature sets of movie 55 using the MLP classifier. The model was trained with a random split of the data and certain movies were chosen to perform prediction, in summary, the training is randomly separated but the test is not.

Analyzing Figure 6 it is possible to see that, as expected, the predictions made by the MLP model are much better, predicting almost everything correctly and, in certain situations, if the model is inaccurate, the probability of confidence that it has is low, having about 60% certainty of the prediction.

If improvements were made to the model predictions by changing isolated predictions, better results could be obtained because there is no likelihood of *fear* in a single second or vice versa. An example of change is to change the model's prediction, for example, *fear* in a single second, to *no-fear*.

5 Deep Learning Classification

In this chapter, the Python library called *Tensorflow* is going to be used in order to try to achieve improved results compared to Chapter 3. First, a baseline Neural Network is created to verify what results are obtained, and then a deeper NN is deployed. Lastly, an RNN is generated using the LSTM layers.

Layer (type)	Output Shape	Param #
Dense 1	(None, 64)	343552
Dense 2	(None, 32)	2080
Dense 3	(None, 32)	1056
Dropout	(None, 32)	0
Dense 4	(None, 1)	33

Total params: 346,721

Fig. 7 Model created using TensorFlow.

5.1 Baseline Neural Network

A Neural Network based on the Batziou et al. 2018 methodology work, containing three hidden layers of 64, 32, 32 neurons with an activation function called relu, and a learning rate equal to 0.001 was deployed using *TensorFlow*. In this case, the NN contain four dense layers and one dropout. The output, last dense layer, is composed of a dense layer with an activation function called sigmoid, which returns only values between 0 and 1. The optimizer used is the Adam and has a loss function called binary cross-entropy. The dropout layer was used with a factor of 0.5, meaning that 50% of the connections will be dropped and whoever enters the dropdown will leave with half of the connections. The structure of this NN can be seen in Figure 7.

A tutorial/code made by TensorFlow TensorFlow Core 2020 was used to classify the imbalanced data. In order not to waste the model's first epochs discovering that the *fear* tag is unlikely, because the *LIRIS-ACCEDE* dataset is imbalanced, the bias was changed using the eq. (1), retrieved from TensorFlow Core 2020.

$$bias = -log_e \frac{samples\, fear}{samples\, no\, fear} \tag{1}$$

A model was trained. This model contains the careful bias calculated, 100 epochs with an early stopping monitoring the recall and a batch size of 4096. The audio and visual features combined were used, as these features sets had the best result with the MLP classifier, scikit-learn algorithm (72.5% F1-score).

Analyzing Figure 8, the train line in recall doesn't change much, from epoch 30, in the training curve, and the validation curve stays in the 0.7 recall. Also, the loss in the validation curve increases, which means that the longer the model is trained, the more loss exist in the validation. In the other hand, the loss in the training curve decrease. It also seems that the model is over-fitting in training since the accuracy, precision and recall curve is almost one. In the other hand, this can occur because the data with the tag *no-fear* is much higher than *fear* tag.

The learned model was then used to perform predictions in the remaining data. This model obtained an F1-score by predicting *fear* of 78.37% and *no-fear* 99.10%.

A ROC (Receiver Operating Characteristics) graph was implemented to see the performance of the model because the ROC curve tells how the model classifies the *fear* tag as *fear* and the *no-fear* tag as *no-fear*.

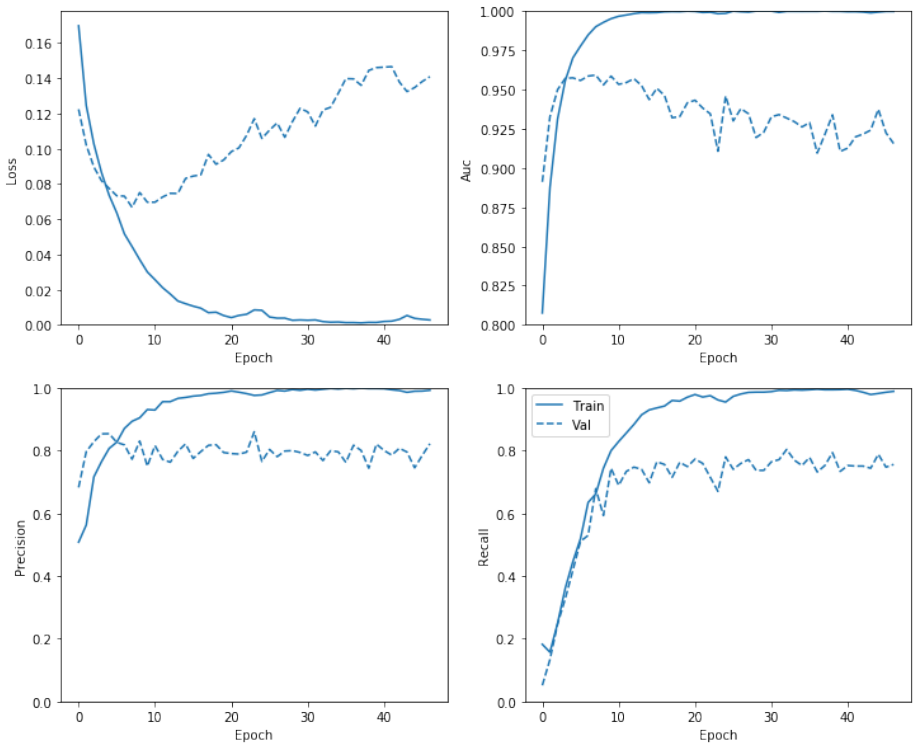

Fig. 8 Variation of the loss, accuracy, precision and recall.

Analyzing the results obtained from the model, in Figure 9, it is possible to see that the AUC in the validation is between 0.5 and 1, so the algorithm can differentiate between *fear* and *no-fear* in certain situations. The AUC in the test is almost 1, which means that the model can distinguish almost all tags and this curve can intend to be an overfit on the train.

Then the same model but this time with different class weights calculated is trained. These weights are calculated because the class *fear* is the one that has fewer samples so needed too have more importance. This may lead the model to pay more attention to examples in the under-represented class TensorFlow Core 2020. The *no-fear* class has a weight of 0.52 and the class of *fear* is 11.83, applying eq.2.

$$weight = \frac{total}{2 * class\,size} \tag{2}$$

Analyzing the graphics in Figure 10 is possible to see that the loss no longer increases. In addition, the recall increases and the precision is worsens, but overfitting seems to continue to exist. The results were proximally close to the baseline model achieve an F1-score by predicting *fear* of 77.86% and *no-fear* 98.91%.

As shown in Table 5, representing the confusion matrix, this model can predict correctly the class *fear* 640 times and 16488 times the *no-fear* class but predict incorrectly 364 times.

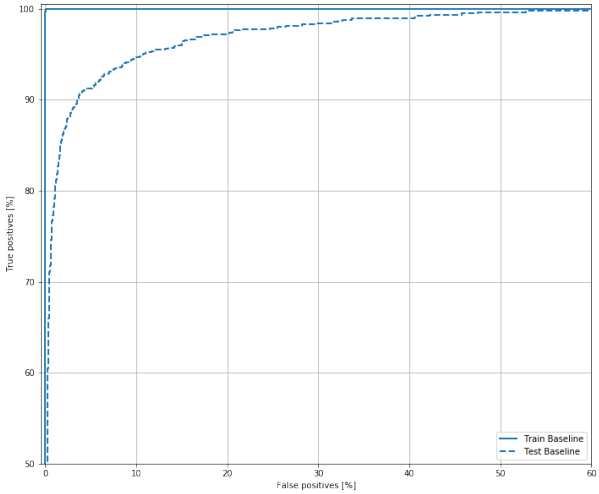

Fig. 9 ROC curve.

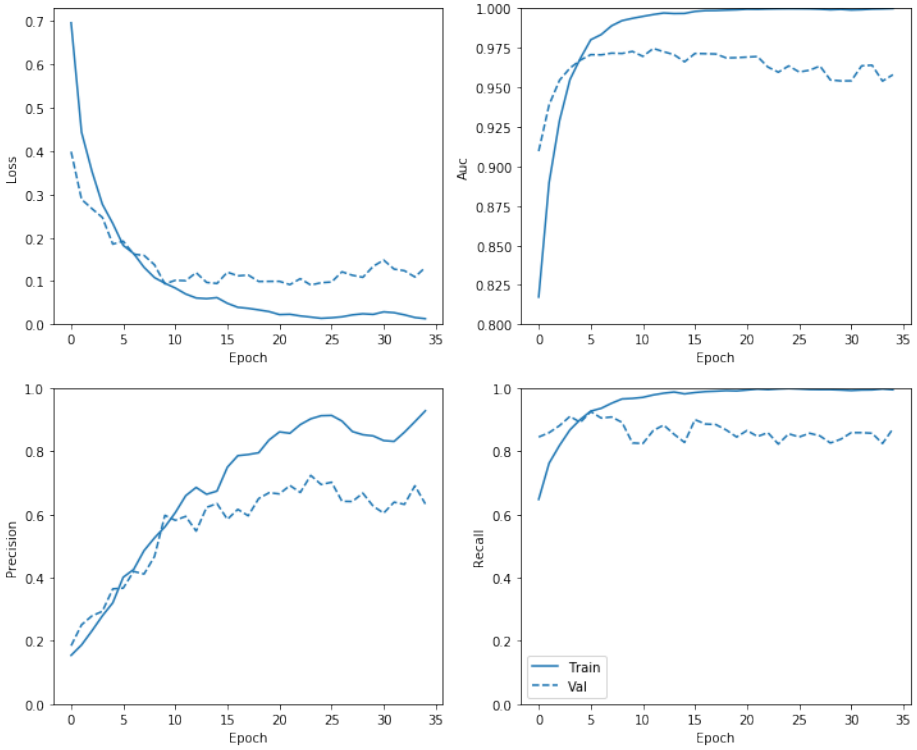

Fig. 10 Number of epochs by the loss, accuracy, precision and recall with class weights.

		Predicted label	
		no-fear	*fear*
Actual label	*no-fear*	16488	255
	fear	109	640

Table 5 Model with class weights.

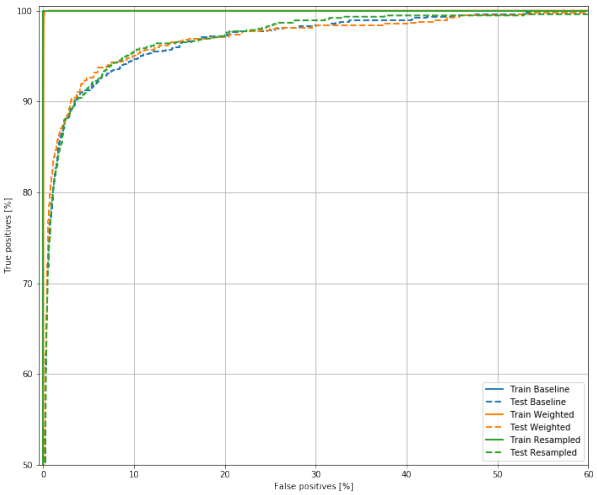

Fig. 11 ROC with all the models.

	F1-score *fear* (%)	**F1-score *no-fear* (%)**	**IoU**	**Time to train (min)**
Baseline model	78.367	99.101	0.644	1.616
Weighted model	77.859	98.908	0.637	1.290
Oversampled model	78.358	99.073	0.644	8.084

Table 6 F1-score using the model with 5 hidden layers.

Lastly, a model with an oversampled data is trained. *Fear* class will be oversampling so that there is more data about *fear*. The model obtain an F1-score by predicting *fear* of 78.36% and *no-fear* 99.07%.

The ROC results for all models are shown in Figure 11. In the end, the study of the ROC graph reveals that all the models obtains the same value of AUC in the testing, and in the training all methods obtain a value close to 1. In testing, the AUC has a value greater than 0.5, which means that the models can differentiate the classes *fear* and *no-fear*.

Table 6 represents the summary of the approaches performed (F1-score and the time to train).

The results in Table 6 produce improved results relative to the MLP scikit-learn algorithm, rising by about 6% in the F1-score of *fear* and 0.3% of *no-fear*. Also, IoU goes from 0.569 to 0.644. It should be noted that the training time has also decreased from 51 minutes to approximately 2.

Even tests using different films to train and test were made. The results were poor using videos features sets, were the best F1-score by predicting *fear* is 8.283%. By forecasting *fear*, the audio features get a 9.687% of F1-score. The *sc* feature, which

	F1-score *fear* (%)	F1-score *no-fear* (%)	IoU	Time to train (min)
Baseline model	79.686	99.231	0.662	1.943
Weighted model	82.694	99.289	0.705	1.787
Oversampled model	84.014	99.336	0.724	18.143

Table 7 F1-score using the model with 10 Dense layers and 9 Dropouts.

uses the Gaussion algorithm to achieve the best result (18.5%), generates a result of 6.904% using this NN algorithm.

5.1.1 Deep Neural Networks

Deeper learning was performed in order to try to achieve better results. Firstly, the visual feature sets are the data utilized and the data used in this section and Section 5.1 were randomly separated so that the train and test data are different in the two sections. In an effort to prevent overfit, the 9 dropouts were included, so the network's complexity has been increased, with 10 dense layers and 9 dropouts (one dense layer, one dropout, one dense layer and so on).

The experiments done in Section 5.1 were also done with this deeper Neural Network (train the model with a calculated bias, also with class weights, and for the last train on the oversampled data). All available data, visual and audio features sets, is used to train the NN, as these are the ones that have achieved the greatest outcomes so far. Table 7 describes the results with the different models.

Using a model with 10 dense layers and 9 dropouts, and oversampled data achieved the best result so far, achieving an F1-score of 84.014% in the class *fear*. The only problem is the training time with the oversampled model that takes 18 minutes to train, but in this situation, if a quicker model is wanted the weighted model is the better one taking only 2 minutes and also obtaining a high F1-score result.

6 Conclusions and Future Work

The main goal of this article was to predict the induction of fear sensations triggered by audiovisual content, using machine learning techniques. The experiments reported here use the *LIRIS-ACCEDE* database, that was previously used in the *MediaEval* benchmarking initiative, in the scope of the *2018 Emotional Impact of Movies Task* challenge. Seven teams competed in this challenge producing outcomes for valence and arousal estimates, and for the forecast of *fear* inducing movie content, sharing similar goals as the ones of this article. The *LIRIS-ACCEDE* database contains several movies, where each segment of one second was annotated as *fear* inducing segment or not.

It has been concluded that the *LIRIS-ACCEDE* dataset, despite being very large, does not contain a significant and representative set of movies for predicting emotions, such as fear. This conclusion is drawn because when having some information about the videos in the training and testing models the results are much better (the best was 84.0%) than those achieved when the models are trained and tested with disjoint film sets. The best result was obtained by combining all the features provided, audio and visual, and trained with a model created with *TensorFlow*, containing 10 dense layers and an extensive application of dropout. Through this model is possible to conclude

that deep learning is more fitting for this type of problem because by increasing the size of the Neural Network from 4 to 10 dense layers, better results are achieved. A simple RNN with 2 LSTM's, a dense layer and a dropout was also used. The models based on LSTMs demanded a significant memory size, as well as long periods of training, which turned out to be a problem considering the existing resources. As a result, the results achieved using such models were not optimized and, despite their potential, they turned out not to be the best performing ones (best result was a 9% F1-score in the *fear* class).

Concerning future directions, efforts should be made for producing larger databases of annotated movies, in order to achieve a more significant content variability. With the current dataset, we can train a model with samples associated to film segments and to correctly predict fear induction in different segments of the same film. However, training with a selection of movies and generalizing predictions for a different selection did not achieve satisfactory results. Another topic that deserves further investigation is the feature extraction process. Current visual features are based on key-frames, and therefore they do not consider temporal movie characteristics, which may also be relevant to trigger emotional states. Additionally, the use of Recurrent Neural Networks should be revisited has more data becomes available.

References

Batziou, Elissavet et al. (2018). "Visual and audio analysis of movies video for emotion detection @ Emotional Impact of Movies task MediaEval 2018". In: *CEUR Workshop Proceedings* 2283.October, pp. 29–31. ISSN: 16130073.

Claude, L Universite and Bernard Lyon (2013). "PHD THESIS D ÂŽ etection des ÂŽ par Rizwan Ahmed KHAN". In.

Dellandréa, Emmanuel et al. (2018). "The MediaEval 2018 emotional impact of Movies task". In: *CEUR Workshop Proceedings* 2283, pp. 6–8. ISSN: 16130073.

Ko, Tobey H. et al. (2018). "Towards learning emotional subspace". In: *CEUR Workshop Proceedings* 2283.3, pp. 4–6. ISSN: 16130073.

Kubat, Miroslav (2017). "An Introduction to Machine Learning". In: *An Introduction to Machine Learning*, pp. 1–348. DOI: 10.1007/978-3-319-63913-0.

Ec-lyon.fr (2015). *LIRIS-ACCEDE*. URL: https://liris-accede.ec-lyon.fr/database.php.

Ma, Ye, Xihao Liang, and Mingxing Xu (2018). "THUHCSI in MediaEval 2018 Emotional Impact of Movies Task". In: *CEUR Workshop Proceedings* 2283. ISSN: 16130073.

Martin Abadi et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: https://www.tensorflow.org/.

Multimediaeval.org (2019). *About MediaEval*. URL: http://www.multimediaeval.org/about/.

Quan, Khanh An C., Vinh Tiep Nguyen, and Minh Triet Tran (2018). "Frame-based evaluation with deep features to predict emotional impact of movies". In: *CEUR Workshop Proceedings* 2283, pp. 2–4. ISSN: 16130073.

SBCoaching (2019). *Emoções: definição, tipos e importância de se ter o controle*. URL: https://www.sbcoaching.com.br/blog/emocoes/.

TensorFlow Core (2020). *Recurrent Neural Networks (RNN) with Keras*. URL: https://www.tensorflow.org/guide/keras/rnn.

Yi, Yun, Hanli Wang, and Qinyu Li (2018). "CNN features for emotional impact of movies task". In: *CEUR Workshop Proceedings* 2283.October, pp. 29–31. ISSN: 16130073.