Applying AI Techniques to Auto-categorise Violent Images

Alexandru Suru

St Hugh's College

University of Oxford

Honour School of Mathematics and Computer Science,

Part C

Supervisors:

Dr. Atilim Gunes Baydin

Prof. Michael Goldsmith

Dr. Harjinder Lallie

Abstract

In digital forensics cases, the analysis of violent images from videos can take a long time for digital investigators. Applications are found in criminal investigations, rating movies or video clips and refereeing incidents during sport events. Artificial intelligence techniques can aid and guide the experts by automating the categorisation and thus speed up the process. This project describes such techniques: decision tree, Bayes classifier, logistic regression and different neural network architectures. Their performance is assessed on a dataset released in 2014, with features from scenes of Hollywood movies and Youtube clips. The dataset contains annotations for violence, based on a subjective definition, and related, more concrete, auditory and visual concepts such as the presence of gunshots or fire in the scenes. The dataset is analyzed in order to explain the relation between violence and the other concepts.

Contents

1 Introduction

Violence detection in images and videos finds applications in criminal investigations, movies and video games rating of appropriateness to certain audiences and arbitration during sport competitions. Automating this process can greatly improve its speed and accuracy. For instance, incidents happening during protests need to be reviewed and visual models can be used to aid the evaluation of the violence present in the footage, according to Donghyeon et al. [13]. There are digital forensics cases in which hours of content have to be studied by security officers in order to identify an event that takes a few seconds [4]. Films may be given ratings based on their content, so that they can be presented as suitable or not, to different age groups. Institutions such as the Motion Picture Association, from the United States or the British Board of Film Classification, from Great Britain provide with such ratings and the level of violence is one of the most decisive factor [35], [1]. The British Board of Film Classification also provides with ratings for video games, while in the USA, it is the Entertainment Software Rating Board. This project is focused on a dataset containing information captured from Hollywood movies and Youtube clips depicting video gameplay, sport incidents and other situations in which violence may take place. The purpose is to explore techniques to auto-categorise scenes as violent or not. Exploration shall consist of presenting the theoretical foundation of the techniques, their implementation and performance analysis.

Given this motivation, it is essential to understand how violence is defined in the first place. According to the Oxford English Dictionary [39] violence means "the exercise of physical force so as to inflict injury on, or cause damage to, persons or property; action or conduct characterized by this; treatment or usage tending to cause bodily injury or forcibly interfering with personal freedom". The World Health Organization [38] offers the following definition: "the intentional use of physical force or power, threatened or actual, against oneself, another person, or against a group or community, that either results in or has a high likelihood of resulting in injury, death, psychological harm, maldevelopment, or deprivation". Note that the second definition focuses on the element of intent. For the purpose of identifying violence in movies, Demarty et al. [8] observed that accidents can also lead to gory scenes. For this reason, Schedl et al. [32] used the following subjective

definition of violent scenes when constructing the dataset: "scenes one would not let an 8-year-old child watch because they contain physical violence". This definition is used to identify violence as part of a variety of instances. Such instances are named concepts and can be auditory or visual. Examples of audio concepts are screams, gunshots, explosions, while fights, fire, gore, blood are some of the visual concepts.

The techniques which are going to be displayed in this project are coming from the artificial intelligence field, more specifically computer vision and machine learning. Artificial intelligence is generally understood to be concerned with developing computer systems that are able to solve tasks which were normally feasible only for humans. It is clear that identifying the presence of violence is a good example of such task. Computer vision is a discipline that studies how computers can solve tasks suitable for human visual systems, by processing images or videos [54]. Machine learning, according to Tom Rainforth [42], focuses on models that make predictions and improve given data. Such models have the potential to reveal patterns which humans were not able to distinguish. Examples of machine learning techniques presented in this project include decision tree, Bayes classifier, logistic regression and neural networks architectures. Decision trees are popular in machine learning and data mining, because of their simplicity and ability to be easily understood and visualized [60]. According to Christopher Bishop [5], using the Bayesian methods has a become mainstream approach to to find patterns and according to Murphy [21] another advantage their interpretability. Logistic regression is a well-known tool for binary classification [36] which will be part of the focus of this project. Neural networks, inspired from biology [26], permit a variety of architectures [52] and are perhaps the most common machine learning approach.

I end this introductory chapter with a roadmap of this report. Chapter 2 provides with a description of the 'Violent Scenes Detection 2014' dataset [32]. I include statistics which I have computed in order to reveal the connection between violence and the other auditory and visual concepts captured in the dataset. Details about dataset cleaning and standardization are also provided. Chapter 3 contains the theoretical background of the techniques and how are they going to be evaluated. Chapter 4 is about how the techniques were implemented and what their results were. Analysis of their performance and comparisons with other benchmarks are found in Chapter 5. Chapter 6, containing

the conclusion, also provides with ways to improve the models presented and hints at alternative approaches.

2 "Violent Scenes Detection 2014" Dataset

I shall refer extensively to the overview paper written by Schedl et. al [32] and to the InterDigital web page [59] associated with the set. The data, produced by Technicolor [59], consists of audio and visual features extracted from movies and video clips from YouTube[1], annotations of the scenes for violence as well as for other related concepts. As mentioned in the Introduction, Chapter 1, scenes containing violence are defined subjectively as scenes "one would not let an 8-year-old child watch because they contain physical violence" [32]. Thus, the Hollywood movies coming from a wide range of genres display a variety of violence levels. For example, "Legally Blond" contains no violence, while in "Saving Private Ryan" a little more than one third of the scenes contain violence, according to Constantin et al. [9]. Copyright issues are the reason extracted features have been provided instead of the actual movies [59]. Two tasks were proposed by the authors of [32] regarding this data: developing algorithms to classify scenes from movies as violent or not, using the given features, and generalizing them so that they could be applied to YouTube video clips, which usually are short in duration.

The dataset is partitioned into 3 subsets: Hollywood-dev, containing features and annotations for 24 movies, to be used for training and validating algorithms; Hollywood-test, containing features and annotations for 7 movies; and YouTube videos with 12 clips with annotations, to be used for the generalization task.

2.1 Features

The data set contains the following visual features:

- CNH = color naming histogram. The authors of [57] define it as a "graphical representation of a distribution of colors within an image. [...] obtained by counting the occurrence of each possible color of the respective color model within the image"
- CM = color moments. Are measures based on color features, used to determine similarity between images [37].

[1] `http://youtube.com`

- LBP = local binary patterns. It is the texture spectrum model proposed in [17] and detailed in [40], [55]

- HOG = histograms of oriented gradients. N. Dalal and B. Triggs [11] present this descriptor, it counts occurrences of gradient orientation in block regions of the image. According to the overview paper of the 'Violent Scenes Dataset 2014' [32], HOG features "exploit the local object appearance and shape within an image via the distribution of edge orientations".

The data set contains the following audio features:

- AE = amplitude envelope, represents a list of amplitude changes of the audio signal.

- RMS = root-mean-square energy. Here, energy means the area under the squared magnitude of the audio signal [47]

- ZCR = zero-crossing rate. Represents the rate at which the audio signal changes its sign [15]

- BER = band energy ratio

- SC = spectral centroid. Determines position of the center of the spectrum.[19]

- BW = frequency bandwidth.

- SF = spectral flux. Is "the spectral change between two successive frames", according to Giannakopoulos et al. [58]

- MFCC = Mel-frequency cepstral coefficients. "Mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound [...] MFCCs are coefficients that collectively make up an MFC", according to [61].

The visual and audio features are provided as bag of words vectors, ordered by frame. The visual features are all multi-dimensional vectors. CM and HOG arrays have 81 entries, CNH arrays: 99 entries and LBP arrays: 144 entries. According to the authors of [32]: CNH arrays map to the following 11 colors: black, blue, brown, gray, green, orange, pink, purple, red, white, and yellow and were compute on 3-by-3 image regions; CM arrays contain the first 3 central moments of an image color distribution: mean, standard

deviation, and skewness, which are computed on 3-by-3 image regions; LBP and HOG were calculated by 3-by-3 spatial division. 8 audio features are present: AE, BW, ZCR, RMS, BER, SC, SF and MFCC. All the corresponding arrays are uni-dimensional with the exception of the MFCC ones, which contain 22 entries for each frame. According to the authors of [32], "the audio exhibits a sampling rate of 44,100 Hz and the videos are encoded with 25 fps".

2.2 Annotations

Apart from the general 'violence' one, there are also annotations for 7 visual and 3 audio concepts in the dev set. As stated in the overview paper [32], all annotations were made by "human assessors in a hierarchical, bottom-up manner, involving two groups of annotators, each consisting of regular annotators and experienced ones". 'Violence' and visual annotations are indexed at frame level, while the audio are indexed by time. According to Sheidl et. al. [32], the total movies duration in the dev set is 50h 3min 12sec, out of which 12.35% are classified as 'violent', the total duration of the movies in the test set is 13h 53min 29 sec, out of which 17.18% are 'violent' while the video clips sum up to 2h 37min 19sec out of which 31.69% are 'violent'.

The 3 audio concepts are: 'explosions', 'gunshots' and 'screams'. The 7 visual concepts are: 'blood', 'carchase', 'coldarms', 'fights', 'fire', 'firearms' and 'gore'. They're annotated to show the presence of the concept and when multiple concepts are present in the same frame, they're marked with 'multiple action scenes'. Some concepts have particular annotations which describe the intensity of the presence of the concept (unnoticeable, low, medium, high in the case of 'blood', for example) or provide with details about how the concept is being manifested ('gruesome skeletons' is an example of such annotations for 'gore'). Because such additional annotations are specific to some movies only (for example, only the movie 'I am legend' has 'group of mutants' annotation for 'gore'), I have only taken into consideration the intensity annotations for 'blood'. The main role of the audio and visual concepts is to provide with more information to the model which ultimately has to classify scenes as violent or not. Thus, the relation between them and violence is of high interest. In order to understand this relation and to improve the understanding of the subjective 'violence' concept, I provide statistics concerning the correlation with the

Blood intensity	Violent frames with blood	Blood frames with violence	Correlation
Unnoticeable	41.04%	37.05%	14.16%
Low	7.97%	77.84%	16.32%
Medium	1.49%	82.63%	7.55%
High	0.26%	95.46%	3.77%

Table 1: Blood concept with intensity level annotations

Concept	Violent frames with concept	Concept frames with violence	Correlation
Explosions	8.83%	53.36%	14.76%
Gunshots	12.36%	56.77%	18.29%
Screams	21.41%	44.69%	19.83%
Carchase	0.25%	13.71%	0.16%
Coldarms	10.2%	18.24%	3.96%
Fights	30.82%	82.61%	38.35%
Fire	15.02%	26.44%	9.57%
Firearms	29.25%	32.7%	17.59%
Gore	16.66%	77.25%	26.75%
Blood	22.9%	45.11%	10.18%

Table 2: Relation between violence and the other concepts

other related concepts in Table 2. Table 1 focuses on the intensity of the 'blood' concept with respect to violence.

In addition to this, here are some interesting facts: over 98.9% of frames containing fights and gore are violent, over 93% of frames containing fights and explosions are violent, over 92.5% of frames containing gore and explosions are violent. The following combinations of 3 concepts imply violence: coldarms & gore & explosions, fire & gore & explosions and more than 99% of all frames containing the following combinations of 3 concepts are violent: coldarms & fights & gore, coldarms & gore & gunshots, fights & fire & gore.

2.3 Cleaning and Standardization

The visual and audio features data requires some cleaning, because a very small number of arrays contain nan (not a number) values. I have observed that most of these occurrences happen in the first and the last arrays and that most of these nan values appear in regions of the arrays dominated by numbers extremely small in absolute value. Hence, I have reassigned these entries to 0, in order to efficiently make use of the data.

Data standardization can bring benefits to training models: it can lead to better accuracy and faster convergence [16]. It consists of subtracting the mean of the dataset

from each element and then divide by the standard deviaion of the data. In the case of multi-dimensional vectors, the mean and standard deviation vectors are computed. Given $x_1, x_2, ..x_n \in \mathbb{R}^d$, the mean vector $M \in \mathbb{R}^d$ satisfies $M_i = \sum_{j=1}^{n} x_{ij}\ \forall i$ and the standard deviation vector $S \in \mathbb{R}^d$ satisfies $S_i = std\{x_{ij}\ | j \in \{1, 2, ..n\}\} = \sqrt{\frac{\sum (x_{ij} - M_i)^2}{n}}$. Hence, the standardized x_i is: $x_i' = (x_i - M)/S$. In order to avoid leakage i.e. accessing data from the test set, I've used the mean and standard deviation vectors of the dev (training) set in order to pre-process the test and generalized datasets as well.

3 Techniques and Methods

I now present the background of the AI techniques that are used later, alongside methods such as smoothing, merging and compressing that will compliment the techniques. In the final part of the chapter I discuss how is the performance of the models measured.

3.1 Decision Tree

A decision tree is a discriminative, supervised technique which can be used for both classification and regression tasks [6]. Each non-leaf node of the tree represents a feature. The edges coming out of a non-leaf node are labeled with the criterion the input satisfies with respect to the feature. Note that this makes decision trees to be easily interpretable. The leafs represent the target output, a class in the case of classification tree or numeric values in the case of regression. Prediction on an input is done by making observations about the item until it reaches a conclusion. It starts from the top of the tree and follows the path given by the decision from the current node until it reaches a leaf, which provides with the prediction. Figure 1, from [6], presents an example of a classification decision tree.

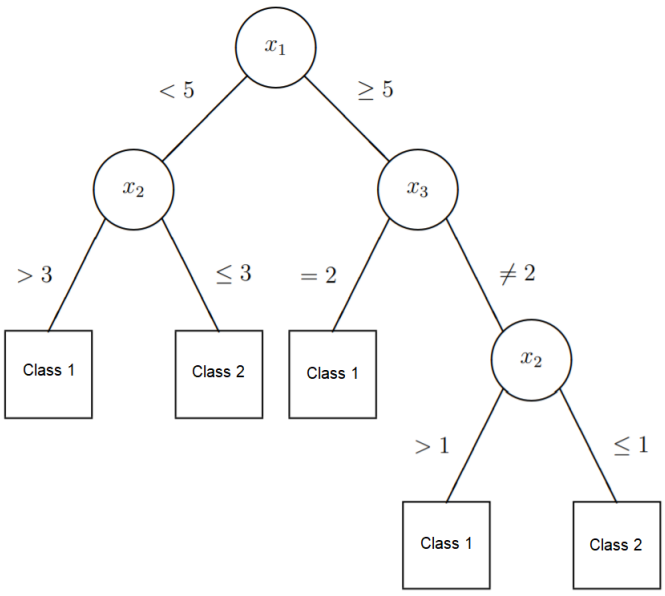

Figure 1: An example of a classification decision tree.

According to Shalev-Shwartz et al. [46], constructing a decision tree can be done by taking the input dataset and then use repeated splits or partitions, based on the

classification criteria. Breiman et al. [6] give the Algorithm 1 for finding a 'good' tree. The algorithms makes use of the notion of purity. Breiman et al. [6] propose *gini* and *entropy* measures in the case of classification trees. Suppose at node i the distribution over, say, k classes is given by the probabilities $p_{i,k}$, which can be estimated using sampling methods. Then $entropy = \sum p_{i,k} \cdot \log(p_{i,k})$ and $gini = \sum_{j \neq k} p_{i,j} \cdot p_{i,k}$.

Algorithm 1: Finding a good decision tree, by Breiman et al.

1. Grow an overly large tree using forward selection. At each step, find the best split. Grow until all terminal nodes either:

(a) have sufficiently few data points

(b) are "pure" i.e. all points in a node have [almost] the same outcome

2. Prune the tree back, creating a nested sequence of trees, decreasing in complexity.

3.2 Naive Bayes Classifier

Naive Bayes Classifier is a generative model, used for supervised learning. According to Murphy [22], in order to solve the classification problem, it first attempts to model the joint distribution of the N x D matrix of input vectors and the corresponding output vector of classes:

$$P(X, Y \mid \theta, \pi) = P(Y \mid \pi) \cdot P(X \mid Y, \theta) = \prod_{i=1}^{N} (P(Y_i|\pi) \cdot P(X_i \mid Y_i, \theta_i))$$

where θ and π define the probability distributions parameters.

Depending on the feature types, different probability distributions can be used for θ. Murphy shows [22] that the marginal distribution P(y | π) can be taken to be the distribution over the classes:

P(y_i = c | π) = $\frac{N_c}{N}$, where N_c = # of input vectors of class c.

The naive assumption is that conditioned on the class label, the input features are independent [22]:

$$P(X_i \mid Y_i, \theta_i) = \prod_{j=1}^{D} P(X_{ij} \mid Y_i, \theta_i), \; for\, all\, 1 \leq i \leq N$$

When classifying a new vector x_{new}, it simply assigns argmax(P(y = c | x_{new}, θ) = argmax(P(y = c) · P(x_{new} | y = c, θ)), where c ∈ {0, 1}

3.3 Logistic Regression

Logistic Regression is a discriminative approach that, according to Murphy [21], models the binary classification problem using the Bernoulli probability distribution as follows: P(y = 1 | x, w) = $\sigma(w \cdot x)$ is called the sigmoid function and P(y = 0 | x, w) = 1 - $\sigma(w \cdot x)$, where x ∈ $\mathbb{R}^D$ is the input vector, $\sigma(t) = \frac{e^t}{1+e^t}$ and w is the model parameter. Thus, the model attempts to compute the value of w, by minimising the value of the binary-cross entropy loss, which is a convex function. Hence, standard convex optimisation methods, such as Newton's method can be applied [23]. For a function to minimise, f : $\mathbb{R}^d \to \mathbb{R}$, the Newton's method consists of iteratively update w_t, until it reaches a satisfactory value, as follows: $w_{t+1} = w_t - H_t^{-1} \cdot g_t$, where H_t is the Hessian of f at w_t and g_t is the gradient of f at w_t. Prediction in Logistic Regression is done by assigning the argmax(P(y = c | x_{new}, w), where c ∈ {0, 1}.

3.4 Dense Neural Networks

Dense neural networks are a type of neural networks built using successive fully-connected perceptrons. According to Murphy [21], the perceptron takes the input, vector X_i , computes its pre-activation value: $z_i = b_i + \sum_j X_{ij} \cdot w_{ij}$, where b_i and w_{ij} are parameters to be learnt, and then applies to this value a given activation function. Some of the most widely used activation functions are the already mentioned sigmoid, $\sigma(\cdot)$, $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, $relu(x) = max(0, x)$. Learning the parameters is done through the backpropagation algorithm [7] and it aims to minimize the values of a given loss function. The algorithm consists of computing the gradient of the loss function in terms of the trainable parameters, by first computing partial derivatives of the loss with respect to the parameters of each layer. It then proceeds with a gradient descend update step. According to Murphy, [21], in general, the gradient descend update rule at step t is: w_t = w_{t-1} - $r_{t-1} \cdot \nabla f(w_{t-1})$, where r_t is the learning rate at step t, and f is the target loss function. Optimizations include which approximating the gradient of the loss on all inputs by computing the average

gradient on only a mini-batch of inputs [21]. For binary classification, the last perceptron can use the sigmoid activation, so that prediction is done by comparing the resulting value to a threshold, usually 0.5.

3.5 Multi-Task Learning

Multi-task learning is defined by Yu Zhang et al. [62] as follows: "Given m learning tasks $\{T_i\}_{i=1}^m$ where all the tasks or a subset of them are related, multi-task learning aims to help improve the learning of a model for T_i by using the knowledge contained in all or some of the m tasks". According to Rich Caruana [44], this can be achieved by having the models that attempt to solve different tasks, share parts of their structure and doing the learning in parallel, so that improvement at solving one task aids the model at solving the others. In the context of this project, the tasks are categorizing scenes for containing or not concepts related to violence. I shall present in the case of neural networks how I applied this technique.

3.6 Long Short-Term Memory Networks

LSTM networks, designed for the first time by Hochreiter et al. [45], are a type of recurrent neural network, meaning they make use of feedback connections in order to capture long-term dependencies in the inputs [3]. LSTMs are able to work over single inputs or sequences of inputs, so they perform well at video classification. According to Hochreiter et al. [45], LSTMs extend the traditional recurrent neural networks with "memory cells", which consist of a neural network layer, a multiplicative input gate and a multiplicative output gate. Each gate contains neural network layers and their activation is usually sigmoid. The role of the input gate is to prevent the cell from being 'disturbed' by irrelevant inputs, while the output gate protects other units from being disturbed by memory of the cell. LSTMs also make use of a forget gate which decides what information from the cell is preserved. According to Gers et al. [14] the forward equations at step t, of the LSTM layer are:

$$\text{forget}_t = \sigma(\text{W}_{forget} \cdot \text{x}_t + \text{U}_{forget} \cdot \text{y}_{t-1} + \text{b}_{forget})$$

$$\text{in}_t = \sigma(\text{W}_{in} \cdot \text{x}_t + \text{U}_{in} \cdot \text{y}_{t-1} + \text{b}_{in})$$

$$\text{out}_t = \sigma(\text{W}_{out} \cdot \text{x}_t + \text{U}_{out} \cdot \text{y}_{t-1} + \text{b}_{out})$$

$z_t = \tanh(W_{cell} \cdot x_t + U_{cell} \cdot y_{t-1} + b_{cell})$

$s_t = forget_t \circ s_{t-1} + in_t \circ z_t$

$y_t = out_t \circ \tanh(s_t)$

where x_t is the input vector and y_t is the output vetor of the LSTM layer; W_{gate}, U_{gate}, b_{gate} are trainable parameters, gate $\in$ {forget, in, out, cell} and $\circ$ is the element-wise product.

3.7 Compression and Decompression

Consecutive frames are likely to be closely related to each other, as being part of the same shot, so concatenating them may prove beneficial to models' learning. I define compression to be a method of data pre-processing as follows: given a window length parameter, take windows of frames or as it is in this case, feature vectors corresponding to the frames, and concatenate them. Compressing the corresponding annotations implies taking a window of annotations and count the proportion of positives. If this proportion is greater or equal to a given threshold, which in this case I set it to be 0.5, then the new annotation for that window is 1, otherwise is 0. When compressing, some frames at the end will remain out of window if the window length is not a divisor of the number of frames. To fix this, one can fill the last window with frames, for example copies of the last one or of the average of those already in window.

To get the predictions of a model that worked over compressed frames, decompression of the predictions is needed. Given the list of predictions and the window length used, decompression means constructing a new list as follows: for each prediction, append to the new list window length copies of the predictions. Note that because of the padding that may take place in the compression, in order for the new list to match the length of the original number of frames, cutting predictions from the end may be required.

3.8 Smoothing and Merging

The proposed models classify frames as violent or not. Obtaining the violent intervals from these classifications is done by finding the sequences of positive frames and returning the start and end indices of each sequence. However, this results in many short intervals which usually do not contribute to the target metric, the MAP2014 score (1). In order to improve

this score, the frames classified by the models, may be further processed before identifying the corresponding intervals. Dai et al.[41] make use of 2 such processes: smoothing and merging.

Smoothing consists of iterating once through windows of frames and counting the percentage of violent frames inside the window. If the percentage is greater or equal to a threshold, all the frames inside the window become positive. Because the movies are encoded with 25 fps, I chose the windows to be of 25 frames. The threshold is 50%.

Merging consists of iterating once through frames and if the distance between any two consecutive violent frames is less or equal to a gap parameter, all the frames between the violent ones become positive. The gap is a hyperparameter and its value should be determined by validation methods.

3.9 Measuring Performance and Validation

In order to describe the performance at classifying frames as 'violent' or not, the following notions and notations are needed:

TP = true positives, number of frames correctly identified as 'violent'; FP = false positives, number of frames wrongly identified as 'violent'; TN = true negatives, number of frames correctly identified as 'non-violent'; FN = false negatives, number of frames wrongly identified as 'non-violent'. Precision = $\frac{TP}{TP+FP}$, Recall = $\frac{TP}{TP+FN}$ and Accuracy = $\frac{TP+TN}{TP+FP+TN+FN}$

Mean average precision measures similarity between the predicted values and the actual ones [33]. Because part of the aim is to also identify the violent intervals, not to just classify frames independently, the authors of [32] suggest using a version of mean average precision, which they call MAP2014, in order to compare models' at this task. The measure considers the predicted violent intervals. If the predicted interval covers more than 50% of a ground truth violent interval, then it is considered a hit. Hits on the same ground truth segment count only as 1, in order not to reward multiple guesses on the same segment. They are not counted as false positives. Thus:

$$MAP2014 = \frac{1}{size(actual\_intervals)} \sum_{i=1}^{size(predicted\_intervals)} precision@i \qquad (1)$$

where

$$precision@i = \frac{\#\,of\ distinct\ hit\ intervals\ until\ the\ i^{th}\ predicted\ interval}{\#\,of\ non-hit\ predicted\ intervals\ until\ the\ i^{th}\ predicted\ interval} \quad (2)$$

Binary-cross entropy is a popular loss function used for training models and measuring performance of binary classifiers. According to Murphy [24], when predicting N points as being 0 or 1, the negative log-likelihood, which is also the binary-cross entropy is:

$$NLL = \frac{-1}{N} \sum_{i=1}^{N} y_i \cdot log(P(y_i = 1)) + (1 - y_i) \cdot log(1 - P(y_i = 1))$$

Validation methods are used for hyperparameter tuning [43]. Two of the most widely used examples are classic validation and k-fold cross-validation. Classic validation consists of inspecting model's performance for different hyperparameter values, on a subset of the training data, called validation set, which was not used for training the model. According to [43], k-fold cross-validation consists of partitioning the training data into k folds and iterating k times the following procedure: each time a new fold is kept as a validation set, the model gets trained on the other k-1 folds and its score on the validation fold is recorded. At the end, the mean of all the scores is the validation score of the model. According to Murphy [25], this method is useful when the training dataset is not large enough for classic validation.

I conclude this section with describing two important notions about measuring models' performance: underfitting and overfitting. According to Murphy, [21], these terms describe the bias-variance trade-off. Underfitting, or high bias, results when an unsuitable model is used to fit the data, which leads to poor performance on both train and test data [21]. This can be avoided by expanding the model's complexity or trying a different approach. Overfitting means high variance. The model performs very well on the train set and poorly on the test set. Ways to deal with overfitting are: reducing model's complexity, applying regularization, early stopping etc. When presenting the proposed models, I specify which of such methods have been used.

4 Implementation and Experiments

The code is publically available on github[2]. The project contains the implementation of the methods and techniques described in Chapter 3 and the development of the statistics from Chapter 2. I provide with functions to process the data: cleaning, standardization, compression and decompression. However, I can not publicize the dataset including the feature vectors. It must be obtained by submitting a request via e-mail to InterDigital[3]. I would like to acknowledge the use of the University of Oxford Advanced Research Computing (ARC) facility[4] in carrying out these experiments. I would also like to express my gratitude towards the support team who assisted me with regard to the logistics.

4.1 Decision Tree

This model uses the Decision Tree Classifier implementation from scikit-learn python library [50]. The standardized feature vectors were merged into 434-dimension input vectors and then were fit together with the corresponding frames annotation into the model. As mentioned in the Section 3.1, *entropy* and *gini* functions can be used to measure the splits. I have opted for using class weights, because while the dataset contains more non-violent frames than violent, the model should be more concerned with identifying the violent ones. The model is highly interpretable: weights for the importance of each of the 434 features are available and the tree structure could be visualized, if not for the large size: it contains 143455 nodes. The project on github contains a file which describes the whole structure of the decision tree.

For validation, k-fold-cross-validation with k = 5 was applied to identify the best performing measure and the value for the gap parameter, used in merging. The validation scores were: *gini* : 10.4%, with gap = 3500 and *entropy* : 12.3% with gap = 3000. The gap values considered were between 0 and 6000, with step size of 100 and the MAP2014 outcomes are shown in Figure 2.

The model was then tested using the similar input vectors from test data. The resulting frames predictions were smoothed and merged. The identified intervals gave a MAP2014

[2] `https://github.com/Cicero17/Applying-AI-Techniques-to-Auto-categorise-Violent-Images`
[3] Details at `https://www.interdigital.com/data_sets/violent-scenes-dataset`
[4] `http://dx.doi.org/10.5281/zenodo.22558`

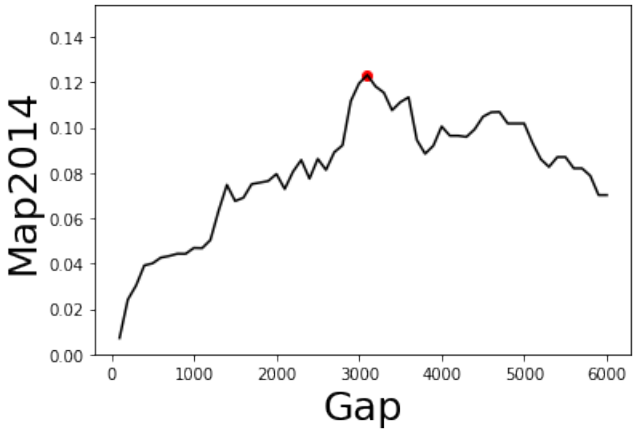

Figure 2: Decision Tree with entropy measure, gap validation plot of 5-fold-cross-validation. Best gap = 3500.

score of 6.0%. On the Youtube video clips dataset, for the generalized task, the model's MAP2014 is 4.6%. Table 3 provides with detailed statistics on the model's performance.

Dataset	Smooth+Merge?	Precision	Recall	Accuracy	MAP2014
dev (training)	no	14.4%	14.9%	78.5%	0%
dev (training)	yes	12.8%	84.3%	27.5%	11.7%
test	no	21.6%	11.3%	77.7%	0%
test	yes	18.2%	44.4%	56.3%	6.0%
Youtube	no	37.0%	24.9%	58.9%	0.3%
Youtube	yes	4.6%	91.8%	39.0%	4.6%

Table 3: Decision tree performance statistics

4.2 Naive Bayes Classifier

This model uses the GaussianNB implementation from scikit-learn python library [51]. Similarly to the previous model, the 434-dimension vectors along with the corresponding violence annotations are used. Each of the 434 entries is considered to be a feature on its own. Note that the 434-dimension vectors do not satisfy the naive assumption: features should be independent conditioned on the output label, but the input vectors were constructed from the audio and visual features of each frame, thus being closely related to each other According to the documentation, [49], they're modelled after the

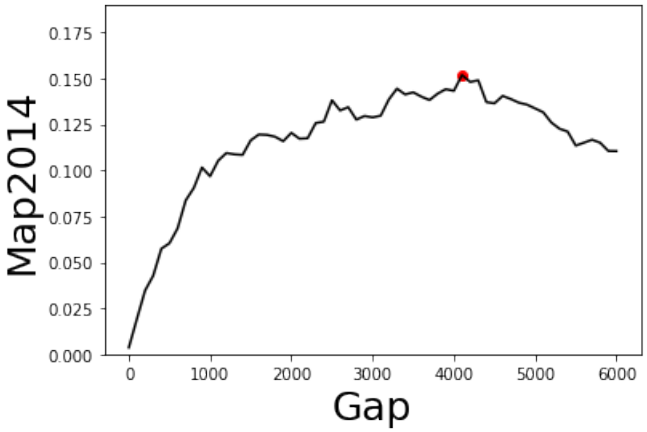

Figure 3: GaussianNB gap validation plot of 5-fold-cross-validation. Best gap = 1100.

Gaussian distribution:

$$P(X_{ij}|Y_i) = \frac{1}{\sqrt{2\pi(\sigma_{Yi})^2}} \cdot e^{-\frac{(X_{ij}-\mu_{Yi})^2}{2\sigma^2_{Yi}}}$$

where μ_{Yi} and σ^2_{Yi} are the mean and the variance of the features from the inputs classified as Y_i

The model was then tested using the similar input vectors from test data. The resulting frames predictions were then smoothed and merged, using gap = 4100. The identified intervals gave a MAP2014 score of 9.0%. The value of the gap hyperparameter was determined via k-fold-cross-validation with k = 5 and the validation MAP2014 score is 15.1%. The gap values considered were between 0 and 6000, with step size of 100 and the MAP2014 outcomes are shown in Figure 3.

On the Youtube video clips dataset, for the generalized task, the model's MAP2014 is negligible: 0.6%, because the intervals with violence from clips are of a much shorter length than the ones from movies and the gap's value of 4100 is too big. However, without smoothing and merging, the result improves: 3%. Figure 3 describes all the components involved.

I conclude this section with more statistics on the model's performance in Table 4.

Dataset	Smooth+Merge?	Precision	Recall	Accuracy	MAP2014
dev (training)	no	21.3%	41.8%	73.7%	0%
dev (training)	yes	13.9%	82.6%	34.8%	10.6%
test	no	28.1%	33.2%	73.9%	0%
test	yes	20.8%	84.4%	42.0%	9.0%
Youtube	no	35.6%	58.1%	48.6%	3%
Youtube	yes	34.9%	100%	35.0%	0.6%

Table 4: Naive Bayes Classifier performance statistics

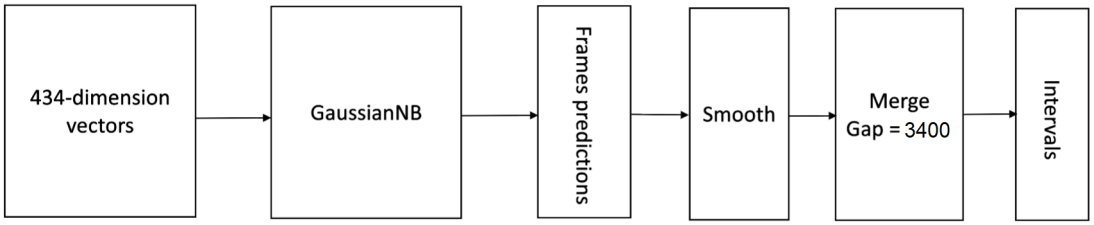

Figure 4: Naive Bayes classifier full architecture

4.3 Logistic Regression

This model uses the Logistic Regression implementation from scikit-learn python library [48]. Using the same approach from the previous model, the standardized feature vectors were merged and fit with the corresponding frames annotation. For training, the model uses l2 regularization and it opts for the class weights to be inversely proportional to the class frequencies. Hence the loss function used is a regularized version of the binary cross-entropy:

$$Loss(Y|X,w) = \frac{-1}{N}\sum_{i=1}^{N}[Y_i \cdot \frac{1}{N_1} \cdot log(\sigma(w \cdot X_i)) + (1-Y_i) \cdot \frac{1}{N_0} \cdot log(1-\sigma(w \cdot X_i))] + \frac{1}{2C}\sum_{i=1}^{D} w_i^2$$

where $\sigma(\cdot)$ is the sigmoid function, N_c = # of input vectors of class c and C is a constant.

Regularization consists of applying a penalty on the model's parameter w and is used to avoid overfitting. The model uses C = 1 and it attempts to minimize the loss by doing at most 30 iterations of gradient descend. Class weights are used to account for the bias in the dataset towards scenes without violence.

Similarly to the previous model, before extracting the violence intervals from the frame predictions, smoothing and merging were applied. In order to determine the value for the

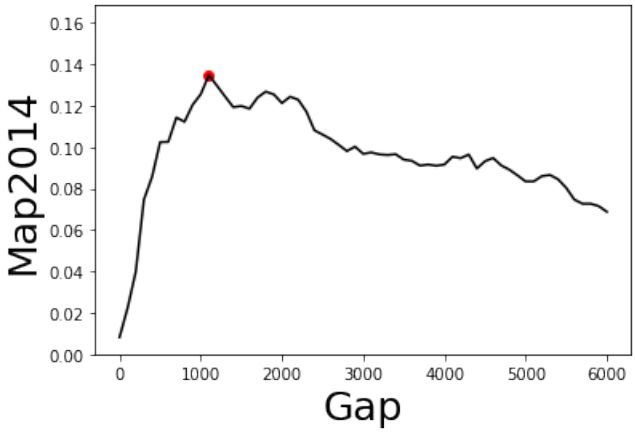

Figure 5: Logistic Regression gap validation plot of 5-fold-cross-validation. Best gap = 1100.

Dataset	Smooth+Merge?	Precision	Recall	Accuracy	MAP2014
dev (training)	no	28.0%	70.7%	73.9%	0.1%
dev (training)	yes	18.2%	98.7%	45.3%	14.1%
test	no	24.1%	42.6%	67.1%	0%
test	yes	21.0%	88.1%	41.2%	12.3%
Youtube	no	32.9%	52.1%	46.2%	0.1%
Youtube	yes	33.8%	86.4%	36.2%	2.9%

Table 5: Logistic Regression performance statistics

gap parameter k-fold-cross-validation with k = 5 was used. The validation MAP2014 score is 13.4% and has been achieved wtih gap = 1100. The gap values considered were between 0 and 6000, with step size of 100 and the MAP2014 outcomes are shown in Figure 5.

The model achieves a MAP2014 score of 12.3% on test and 2.9% on Youtube clips dataset. I present detailed statistics on model's performance in Table 5.

4.4 Neural Networks

These models make use of the tensorflow.keras python module [56], in particular the Sequential model [29] with layers [28]. Custom layers can be created and models may make use of them.

4.4.1 Dense Neural Networks

The layers used by such neural networks are dense layers, with different activation functions, and dropout layers which take a rate $\in [0, 1]$, according to the documentation [27]. During training, at each gradient update step, dropout is applied: a fraction, equal to the rate, of units are uniformly selected; they will not be updated. The purpose of the dropout layer is to tackle overfitting.

The next architecture proved to improve previous results. I gave it the name 'DNN 0', because I shall refer to it later. It is presented in Figure 6 and it consists of: consecutive dense layers with tanh activation function as follows: 1 of size, i.e. number of perceptrons 434, then 2 of size 256, 2 of size 128, 2 of size 64, 2 of size 32, 2 of size 16, 1 of size 4, a dropout layer with rate 0.5 and a final dense layer, of size 1 with sigmoid activation function. The architecture performed better with tanh as activation for the non-final dense layers than when using relu. The model's loss function is binary cross-entropy and is the two classes are weighted inversely proportional to the size of the classes. The adam optimizer [2], [30] achieved better performance than the stochastic gradient descent sgd optimizer [53]. The 434-dimension input vectors along with their corresponding violence annotations are fed to the model. Using validation on 20% of data, by focusing on best precision and recall values, I determined that when training on batches of size 2000 for 10 epochs the dense network performs best. The model classifies frames separately, so again smoothing and merging have been applied before extracting the violence intervals. The gap parameter for merging was determined by k-fold cross-validation on training data, with k = 5 and MAP2014 was the metric used. The gap values varied between 0 and 3000 with a step of 100 and the best validation score was achieved for value 1600. The resulting model achieves a MAP2014 score of 14% on test and 2.6% on Youtube clips dataset. I provide further details on its performance in Table 6.

It is worth exploring using dense neural networks on compressed data, because consecutive frames can be closely related to each other. I have run experiments on the data compressed by windows of length 3, 5 and 25. While the best model working on data compressed by 5 achieved a MAP2014 score of 13.4% and all models working on data compressed by 25 performed very poorly, achieving MAP2014 scores of less than 2%, a

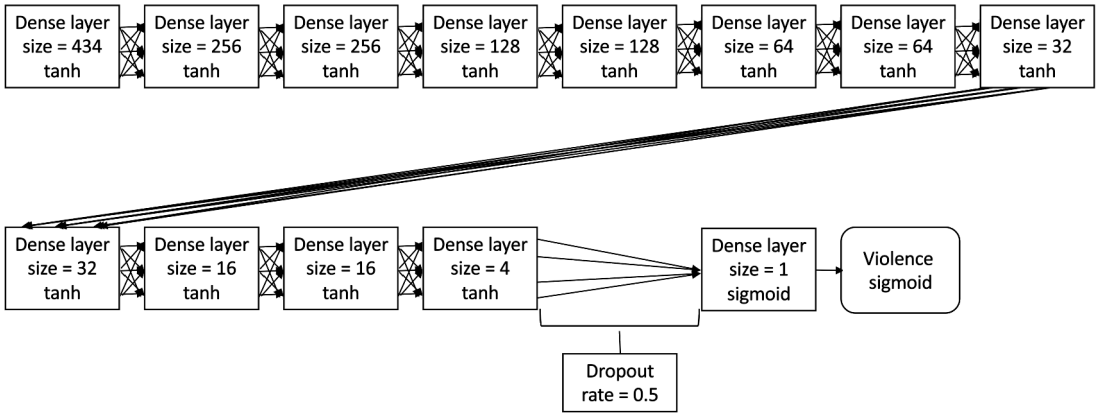

Figure 6: DNN 0 structure

Dataset	Smooth+Merge?	Precision	Recall	Accuracy	MAP2014
dev (training)	no	26.1%	80.6%	69.4%	0%
dev (training)	yes	15%	99.6%	30.5%	11.5%
test	no	24.6%	38.2%	69.2%	0%
test	yes	20.6%	93.8%	36.9%	14%
Youtube	no	33.4%	53.3%	46.5%	0.1%
Youtube	yes	34.3%	92%	35.8%	2.6%

Table 6: DNN 0 performance statistics

model working on data compressed by 3 achieved a slightly better performance than the previous dense neural network, DNN 0. I provide details about this last model.

The architecture is almost same as that of DNN 0, with the single exception of the first dense layer, its size being 1302 instead of 434. This design choice gave better results when working on compressed data. Training is done in batches of size 2000 for 10 epochs, the loss function is the binary cross-entropy with weights accounting for class sizes.

The model ran on compressed data, with the windows length being 3, so the input vectors have size 1302. Padding was done by filling the last window with copies of the last frame. The resulting predictions of the compressed frames are then decompressed. Finally, smoothing and merging is applied before extracting the intervals. The gap hyper-parameter from merging was classically validated on 20% of the training data with values ranging from 0 to 6000, steps of 100 and the value that gave the best MAP2014 score was 4200. The model achieved a MAP2014 score of 14.4% on test and 1.9% on Youtube clips dataset. I now present more details about its performance. A drawing of the whole structure is Figure 7.

Dataset	Smooth+Merge?	Precision	Recall	Accuracy	MAP2014
dev (training)	no	30.6%	52%	79.5%	0.3%
dev (training)	yes	17.4%	96.1%	43.4%	12.6%
test	no	26.8%	27%	74.7%	0%
test	yes	22%	91%	43.3%	14.4%
Youtube	no	32.5%	53.6%	44.9%	0.4%
Youtube	yes	34.1%	93.5%	34.8%	1.9%

Table 7: DNN on compressed data performance statistics

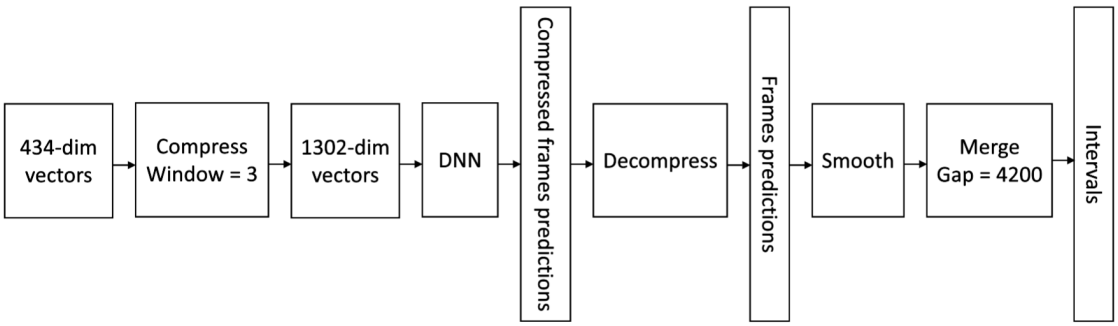

Figure 7: Dense neural network on compressed data architecture

4.4.2 Multi-Task Learning

In this section I present an architecture that applies multi-task learning in order to create new feature vectors. The goal is to augment the violence predictions made by the first dense neural network from the previous Subsection 4.4.1. Because the full architecture is rather complex, involving multiple dense neural networks. Figure 8 is the scheme of the architecture and Figure 9 presents the frames violence predictions post-processing. I give first an overview of the architecture and then details about each of its component. Statistics on performance are at the end of the section.

The model consists of 3 initial neural networks, 2 of which implement multi-task learning, that are used to produce a 73 dimension feature vectors. These vectors are then processed by a final dense neural network that gives the final violence frames predictions. The frames predictions are then smoothed, merged (gap parameter = 1700, obtained by classic

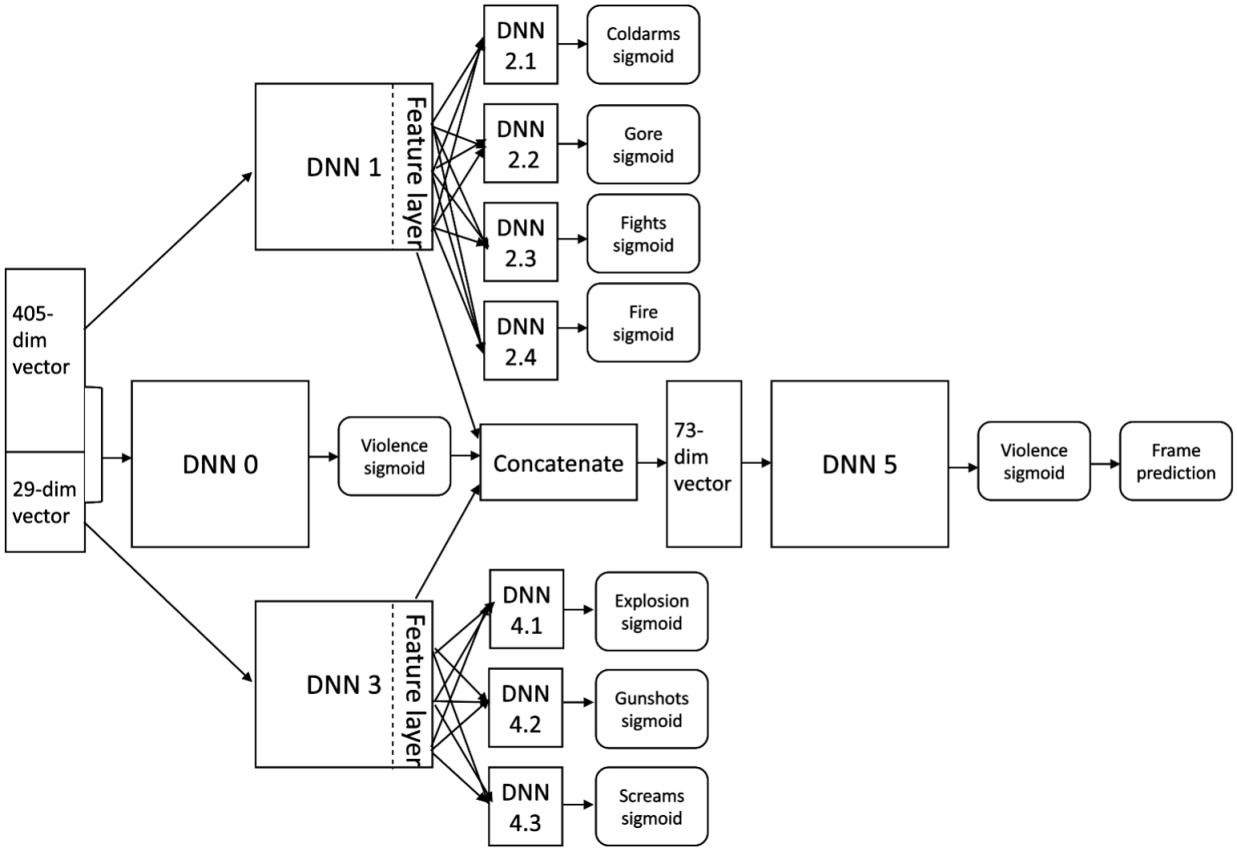

Figure 8: Neural networks including multi-task learning structure

validation on 20% of the intermediate feature vectors) and then the violence intervals are extracted. I now proceed with descriptions of each component.

DNN 0 is the first dense neural network described from the Subsection 4.4.1. It runs on the full 434 input vector and it outputs the probability that the corresponding frame is violent, i.e. the value of the sigmoid function in from its last layer. In the Subsection 4.4.1 this value is used to compute the final prediction, by comparing against the threshold of 0.5. In this case, however, the sigmoid value is kept to be part of the 73 feature vector that is later used by DNN 5.

DNN 1 and DNN 2.1, DNN 2.2, DNN 2.3, DNN 2.4 are part of the same neural network which applies multi-task learning. The tasks are predicting the presence of 4 specific visual concepts that related to violence. The visual concepts that it predicts are 'coldarms', 'gore', 'fights' and 'fire'. I selected these 4 concepts because of their relation to violence. Statistics and observations describing this relation are found at the end of the dataset description, Chapter 2. The neural network runs on vectors of 405 dimension,

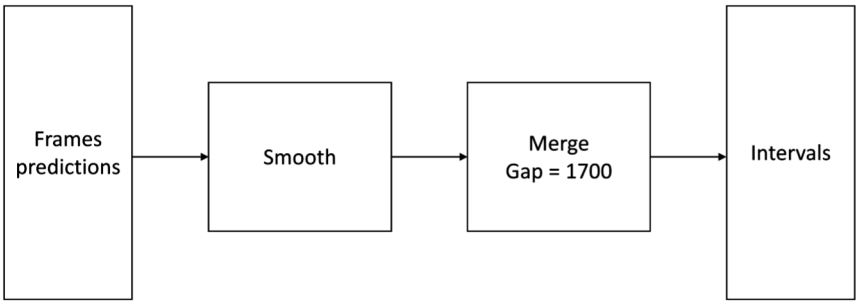

Figure 9: Multi-task learning model predictions post-processing

obtained from the visual features: CNH, CM, LBP and HOG. The network consists of shared dense layers, which form DNN 1, and task specific layers. DNN1 1's architecture is the following: consecutive dense layers, all with tanh activation function, of sizes 405, 256, 256, 128, 128, 64. The last layer is considered to be the feature layer and its output is also used to build the 73 dimension feature layer. The task specific parts, DNN 2.1, DNN 2.2, DNN 2.3, DNN 2.4, have the same architecture: dense layers with tanh activation of sizes 64, 32, 16, 8, 4 followed by a dropout layer with rate 0.5 and a final dense layer of size 1 and sigmoid activation. Training of the whole neural network uses the input vectors along with the concepts annotations for the corresponding frames. It is done for 10 epochs, in batches of size 2000 via a custom multi-loss function. The multi-loss function is a weighted sum of binary cross-entropy loss values for each concept. In the context of this model, all weights are equal.

DNN 3 and DNN 4.1, DNN 4.2, DNN 4.3 are part a neural network similar to the one described above. It also implements multi-task learning, the concepts involved are auditory: 'explosions', 'gunshots' and 'screams'. Again, the selection was based on the statistics on the violence correlation, presented in the Chapter 2. Note that together with the other considered concepts, many subsets containing some of them imply violence. The neural network inputs are 25 vectors, made from the audio features: AE, RMS, ZCR, BER, SC, BW, SF and MFCC. The network consists of shared dense layers forming DNN 3 and task specific dense layers. DNN 3's architecture is the following: fully-connected layers with tanh activation of sizes: 29, 16, 8, 8. The last layer is the feature layer and its outputs are also used for the 73 dimension feature vectors. DNN 4.1, DNN 4.2, DNN 4.3

Dataset	Smooth+Merge?	Precision	Recall	Accuracy	MAP2014
dev (training)	no	48.8%	86.5%	87.1%	0%
dev (training)	yes	22.9%	99.5%	58.7%	14.3%
test	no	24.5%	34.3%	70.5%	0%
test	yes	21.2%	92.8%	39.5%	15.9%
Youtube	no	34.9%	99.9%	35.0%	10.3%
Youtube	yes	34.9%	100%	34.9%	0.6%

Table 8: Multi-task learning architecture performance statistics

have the same architecture: dense layers with tanh activation of sizes 4, 4, 2, 2; followed by a dropout layer with rate 0.5 and a final fully connected single neuron with sigmoid activation. Training of the whole neural network is done for 10 epochs, in batches of size 2000 via the multi-loss function.

DNN 5 is the dense neural network that works over the intermediate feature vector. Its architecture is: fully connected layers with relu activation of sizes: 73, 64, 64, 32, 32, 16, 16, 8, 8, 4; followed by a dropout layer with rate 0.5 and a final dense layer of size 1, with sigmoid activation. Training is done in 10 epochs, in batches of size 2000 via the binary cross-entropy loss.

The whole architecture achieved a MAP2014 score of 15.9% on test and 0.6% on Youtube clips dataset. An important observation: the architecture performed much better on Youtube clips dataset without smoothing and merging, achieving a MAP2014 of 10.3%. I provide with complete statistics in Table 8.

4.4.3 Long Short-Term Memory Networks

This model uses the LSTM layer with forget gate implemented in tensorflow keras [31]. The model works over sequences of 3 input vectors, where the vectors are the 73-dim feature vectors, obtained via applying multi-task learning described in the previous Subsection 4.4.2 and Figure 8. Padding with copies of the last vector can be applied for the last sequence. Each feature vector corresponds to a frame and so it has a violence annotation. Sequences have a corresponding violence annotation given by the majority of the violence annotations of the vectors in the sequence. Thus, the sequences annotations are the compressed annotations when the window parameter is 3. To recover frame level

Dataset	Smooth+Merge?	Precision	Recall	Accuracy	MAP2014
dev (training)	no	48.4%	93.3%	86.9%	0.1%
dev (training)	yes	16.5%	99.9%	37.9%	12.2%
test	no	26%	20.2%	76.4%	0%
test	yes	20.7%	90.9%	38.7%	17.4%
Youtube	no	35.2%	85.9%	39.9%	0.1%
Youtube	yes	34.9%	100%	34.9%	0.6%

Table 9: LSTM Neural Network performance statistics

annotation predictions, the sequence predictions are decompressed. Experiments with sequences of length 5 and 25 were conducted, but the best performance was achieved with length 3, similar to the results of dense neural networks work on compressed data from Subsection 4.4.1.

The model's architecture consists of both LSTM and dense layers, as this hybrid approach is the one that improves previous results. The model is as follows: consecutive LSTM layers with tanh activation, which return the state output for each member of the sequence, of sizes: 73, 64, 64, 32; a LSTM layer with output for the whole sequence with tanh activation of size 32; dense layers with tanh activation of sizes: 32, 32, 16, 16, 4; a dropout layer with rate 0.5 and a final dense layer of size 1, with sigmoid activation. Training is done in batches of 2000, for 10 epochs and the binary cross-entropy loss function is weighted to account for class imbalances. The frame level violence predictions are then smoothed and merged with gap parameter 3200 before extracting the intervals. Gap parameter's value was obtained by classic validation on 20% of the sequences. The model achieved a MAP2014 score of 17.4% on test and 0.6% on Youtube clips. It is worth noting that with only smoothing as post-processing, the model's MAP2014 score on Youtube clips is 8.3%. In the Table 9 are the full statistics on this model's performance.

5 Analysis

I discuss the presented models' performance, including limitations, on the test and Youtube clips datasets. For comparison, I shall refer to the following papers presenting benchmarks for the violence detection task in Hollywood movies: from 2012 [12] and from 2020 [9].

The model that achieved the lowest MAP2014 score, on test, with smoothing and merging is the Decision Tree, 6.0%. However, it is worth pointing out that without post-processing, it is the model that achieves the best test accuracy, 77.7%.

The Naive Bayes Classifier without smoothing and merging has the best precision on test, 28.1%, but the accuracy is mediocre, 73%, which is somewhat surprising, given that it is the only model that does not use class weights. The fact that the features do not satisfy the naive assumptions seems to had an impact on the performance of the model.

Best recall on test, 93.8% is achieved by the DNN 0 model, with smoothing and merging included. In fact, all of the neural networks tend to achieve good recall scores. One key observation is that recall seems to be the measure which correlates best with the MAP2014.

Best MAP2014 score on test was 17.4%, achieved by the LSTM neural network from Subsection 4.4.3 with data post-processing. Best MAP2014 score on Youtube clips was 10.3% given by the DNN 5 model from Subsection 4.4.2, without data post-processing. Both of these models worked over the intermediate 73-dimension feature vectors described in Subsection 4.4.2. It is clear that applying the multi-task technique had a positive effect to the models. This is because the tasks were identifying concepts correlated to violence, thus augmenting the models' performance in the general task.

A discussion about post-processing is in order. In the case of all models, smoothing and merging were applied before extracting the violence intervals. Note that the effectiveness of the smoothing and merging is a consequence of attempting to minimize the number of false negatives; which also leads to a great improvement in the recall. Both operations aim to force more robust sequences of frames classified as violent. This results in a trade-off between accuracy, due to the increase in the number of false positives, and recall. Out of the 2 operations, merging is the one with a more drastic impact. This can be seen in the very poor performance of the models on the Youtube clips, compared to the movies

datasets. This is due to the fact that violent scenes in clips are much shorter than the ones in movies. Thus, the large values of the gap parameter computed by validation on movies do not fit well on the clips dataset.

5.1 Benchmark Comparison

The benchmark from 2012 [12] presents the participants results at the violence detection task on a subset of the Hollywood movies from dev. Participants used movies scenes to train, not preselected feature vectors. Evaluation was done in terms of a cost function defined as a weighted sum of the number of false positives and false negatives, with a bias ratio of 1:10; and the classic version of mean average precision metric, MAP. This metric was changed to MAP2014 version, because the classic MAP counts multiple hits on the same violence interval. Schedl et al. [32] explain the new metric "avoids the problem that participating teams could artificially increase their algorithms' MAP scores by predicting more, but shorter segments within the boundaries of a violent scene". Despite this, the models presented in this project achieve performance comparable to the one described in [12]. Only 1 model, the one proposed by TI: joint participation Technicolor-INRIA, achieved a higher MAP score, of 25% at violent interval detection in movies, than the LSTM from Subsection 4.4.3. All the others gave classic MAP scores of less or equal to 17%.

The benchmark published in year 2020 [9] reviews extensively the improvements of the dataset and models' performance. The new dataset is named VSD96, containing more than 96 hours of videos features. Compared to the 'Violent Scenes Detection 2014' dataset which was used in this project, VSD96 has all the Hollywood movies, from both dev and test sets, for training purposes and testing is done on Hollywood-like movie clips. An additional objective definition for violence is introduced for annotation: "physical violence or accident resulting in human injury or pain". The paper reveals the proposed models' results at different stages of the dataset. For comparison, I refer to the "MediaEval 2014" stage. The best result was achieved by Dei et al. [41], a MAP2014 score of 63% with a deep neural network model, working over additional features, not part of the given dataset, extracted directly from the movies, such as space-time interest points, STIP [18] and histograms of optical flow, HOF [34]. Figure 10, which is cropped from Fig. 3 from

[12], presents additional details about the models' performance from the benchmark and can be used to compare with the LSTM's performance of 17.4%. It is clear that the models presented in this project don't match the requirements of a commercial product.

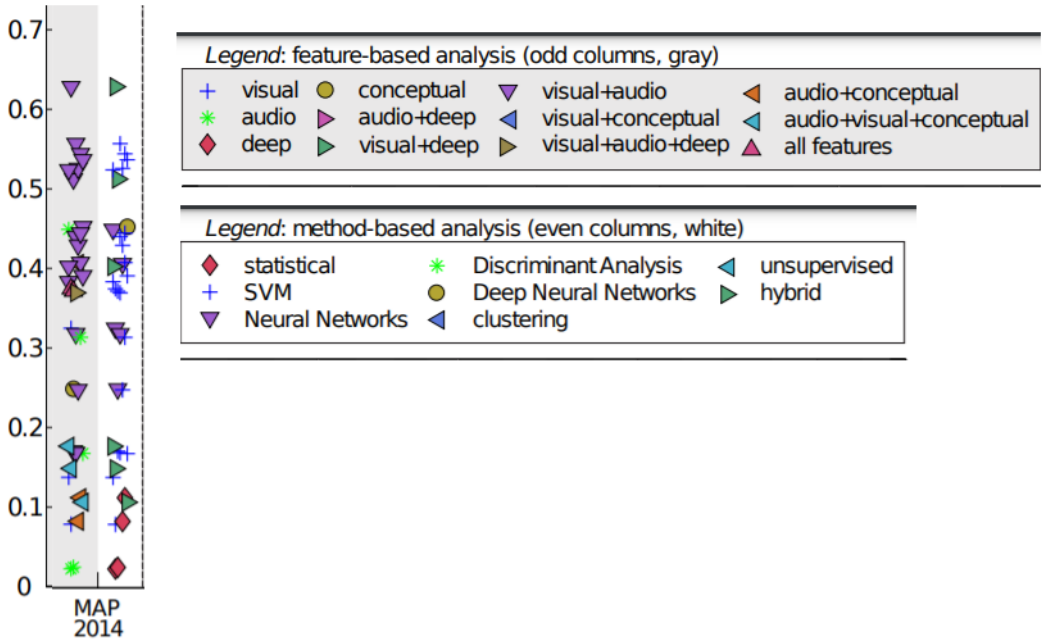

Figure 10: Benchmark statistics. The LSTM based model achieved a MAP2014 score of 0.174. Note that many of the models in the benchmark made use of features outside the dataset, developed directly from the movie images.

6 Conclusion and Future Work

Detection of violence in images or videos is a challenging task which finds applications in many contexts. This project explored a wide range of artificial intelligence techniques in order to identify violence in a collection of Hollywood movies scenes and Youtube video clips. Violence was given a subjective, more general, definition. For a better understanding and to improve the presented models, its relation to other more concrete, related concepts was revealed through the development of relevant statistics. This relation was also used as part of the multi-task learning approach, which lead to an improvement in performance. The A.I. techniques presented were: Decision Tree, Naive Bayes Classification, Logistic Regression and Neural Networks architectures, including LSTM layers. Techniques' theoretical background was described in detail and their implementation performance was analysed.

Future work involving the 2014 stage of the 'Violent Scenes Detection' dataset [32] may focus on more experiments with multi-task learning, which proved to improve the performance of the models focusing initially only on the violence detection. Such experiments may consist of selecting other related concepts as multi-tasks and developing other feature vectors. Other techniques can also be tried, such as support vector machine [10].

Additional work can be done using features outside the ones given in the dataset, as revealed by Dai et al. [41]. However, this would require direct access to the movies scenes and not via pre-selected features. Experiments with different collections of features, the application of feature selection algorithms and ways to encode the features, such as Fisher Vectors [20], may improve the results given by the presented techniques.

References

[1] URL: https://www.bbfc.co.uk/about-classification/ratings-info. (accessed: 24.02.2021).

[2] *Adam Optimizer*. URL: https://keras.io/api/optimizers/adam/. (accessed: 09.01.2021).

[3] Schaefer Anton Maximilian, Udluft Steffen, and Zimmermann Hans-Georg. "Learning long-term dependencies with recurrent neural networks". In: *Neurocomputing* 71.13 (2008), pp. 2481–2488.

[4] Miriana Bianculli et al. "A dataset for automatic violence detection in videos". In: *Data in Brief* 33 (2020), p. 106587. ISSN: 2352-3409. DOI: https://doi.org/10.1016/j.dib.2020.106587. URL: https://www.sciencedirect.com/science/article/pii/S2352340920314682.

[5] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[6] L. Breiman et al. *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984.

[7] Buscema and Massimo. "Back Propagation Neural Networks". In: *Substance use and misuse* 33 (1998).

[8] Andrea Fusiello, Vittorio Murino, and Rita Cucchiara, eds. *A Benchmarking Campaign for the Multimodal Detection of Violent Scenes in Movies*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 416–425. ISBN: 978-3-642-33885-4.

[9] M. G. Constantin et al. "Affect in Multimedia: Benchmarking Violent Scenes Detection". In: *IEEE Transactions on Affective Computing* (2020), pp. 1–1.

[10] Nello Cristianini and Elisa Ricci. "Support Vector Machines". In: *Encyclopedia of Algorithms*. Ed. by Ming-Yang Kao. Boston, MA: Springer US, 2008, pp. 928–932. URL: https://doi.org/10.1007/978-0-387-30162-4_415.

[11] Navneet Dalal and Bill Triggs. “Histograms of Oriented Gradients for Human Detection”. In: *IEEE Computer Society* 1 (June 2005). Ed. by Cordelia Schmid, Stefano Soatto, and Carlo Tomasi, pp. 886–893. DOI: 10.1109/CVPR.2005.177. URL: https://hal.inria.fr/inria-00548512.

[12] Claire-Hélène Demarty et al. “A Benchmarking Campaign for the Multimodal Detection of Violent Scenes in Movies”. In: vol. 7585. Oct. 2012, pp. 416–425.

[13] Won Donghyeon, Steinert-Threlkeld Zachary C., and Joo Jungseock. “Protest Activity Detection and Perceived Violence Estimation from Social Media Images”. In: *Proceedings of the 25th ACM International Conference on Multimedia 2017* (2017).

[14] Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. “Learning to Forget: Continual Prediction with LSTM”. In: *Neural Computation* 12.10 (2000), pp. 2451–2471. URL: https://doi.org/10.1162/089976600300015015.

[15] Fabien Gouyon, Francois Pachet, and Olivier Delerue. “On the Use of Zero-Crossing Rate for an Application of Classification of Percussive Sounds”. In: (Aug. 2002).

[16] Joel Grus. *Data Science from Scratch.* Sebastopol, CA : O’Reilly Media, 2015.

[17] Dong-chen He and Li Wang. “Texture Unit, Texture Spectrum, And Texture Analysis”. In: *IEEE Transactions on Geoscience and Remote Sensing* 28.4 (1990), pp. 509–512. DOI: 10.1109/TGRS.1990.572934.

[18] Laptev I. “On space-time interest points”. In: *Int J Comput Vision* (2005).

[19] Grey J. M. and Gordon J. W. “Perceptual effects of spectral modifications on musical timbres”. In: *Journal of the Acoustical Society of America* (1978).

[20] Sanchez Jorge et al. “Image Classification with the Fisher Vector: Theory and Practice.” In: *International Journal of Computer Vision Springer Verlag* (2013), pp. 222–245.

[21] Murphy K. P. “Machine Learning: A Probabilistic Perspective”. In: MIT Press, 2012.

[22] Murphy K. P. “Machine Learning: A Probabilistic Perspective”. In: MIT Press, 2012. Chap. 3.

[23] Murphy K. P. “Machine Learning: A Probabilistic Perspective”. In: MIT Press, 2012. Chap. 8.

[24] Murphy K. P. “Machine Learning: A Probabilistic Perspective”. In: MIT Press, 2012. Chap. 4.

[25] Murphy K. P. “Machine Learning: A Probabilistic Perspective”. In: MIT Press, 2012.

[26] N.L.W. Keijsers. “Neural Networks”. In: (2010). Ed. by Katie Kompoliti and Leo Verhagen Metman, pp. 257–259. DOI: https://doi.org/10.1016/B978-0-12-374105-9.00493-7. URL: https://www.sciencedirect.com/science/article/pii/B9780123741059004937.

[27] *Keras Dropout*. URL: https://keras.io/api/layers/regularization_layers/dropout/. (accessed: 09.01.2021).

[28] *Keras Layer*. URL: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Layer. (accessed: 03.10.2021).

[29] *Keras Sequential Model*. URL: https://www.tensorflow.org/api_docs/python/tf/keras/Sequential. (accessed: 03.10.2021).

[30] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *arXiv e-prints*, arXiv:1412.6980 (Dec. 2014), arXiv:1412.6980. arXiv: 1412.6980 [cs.LG].

[31] *LSTM Layer Tensorflow*. URL: https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM. (accessed: 13.02.2021).

[32] Schedl Markus et al. “VSD2014: A Dataset for Violent Scenes Detection in Hollywood Movies and Web Videos”. In: *International Workshop on Content-Based Multimedia Indexing* (2015).

[33] *Mean Average Precision*. URL: https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52. (accessed: 18.11.2020).

[34] Rensso Mora, Carlos Caetano, and William Schwartz. “Histograms of Optical Flow Orientation and Magnitude to Detect Anomalous Events in Videos”. In: Aug. 2015.

[35] Inc Motion Picture Association. *CLASSIFICATION AND RATING RULES*. 2020.

[36] Marlene Müller. "Generalized Linear Models". In: *XploRe — Learning Guide*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 205–228. ISBN: 978-3-642-60232-0. DOI: 10.1007/978-3-642-60232-0_7. URL: https://doi.org/10.1007/978-3-642-60232-0_7.

[37] Keen Noah. "Color Moments". In: (Feb. 2005). URL: https://www.researchgate.net/publication/308953685_Color_moments.

[38] World Health Organization. *World report on violence and health: summary*. 2002.

[39] *Oxford English Dictionary*. 2nd ed. Oxford University Press, 1989.

[40] "Performance evaluation of texture measures with classification based on Kullback discrimination of distributions". In: *12th IAPR International Conference on Pattern Recognition* 1 (1994), pp. 582–585.

[41] Dai Qi et al. "Fudan-NJUST at MediaEval 2014: Violent Scenes Detection Using Deep Neural Networks". In: *CEUR Workshop Proceedings* (2014).

[42] Tom Rainforth. *Advanced Topics in Machine Learning: Bayesian Machine Learning*. Department of Computer Science, Oxford, 2020.

[43] Sebastian Raschka. *Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning*. 2020. arXiv: 1811.12808 [cs.LG].

[44] Caruana Rich. "Multitask Learning". In: *School of Computer Science, Carnegie Mellon University, Pittsburg* (1997).

[45] Hochreiter Sepp and Schmidhuber Jürgen. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[46] Shai Shalev-Shwartz and Shai Ben-David. "Decision Trees". In: *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014, pp. 212–218. DOI: 10.1017/CBO9781107298019.019.

[47] *Signal Energy*. URL: https://www.sciencedirect.com/topics/engineering/signal-energy. (accessed: 04.07.2020).

[48] *sklearn LogisticRegression*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. (accessed: 26.12.2020).

[49] *sklearn.naive-bayes*. URL: https://scikit-learn.org/stable/modules/naive_bayes.html. (accessed: 16.11.2020).

[50] *sklearn.naive-bayes.DecisionTreeClassifier*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html. (accessed: 15.11.2020).

[51] *sklearn.naive-bayes.GaussianNB*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html. (accessed: 16.11.2020).

[52] Bas van Stein, Hao Wang, and Thomas Bäck. "Neural Network Design: Learning from Neural Architecture Search". In: (2020). arXiv: 2011.00521 [cs.LG].

[53] *Stochastic Gradient Descend Optimizer*. URL: https://keras.io/api/optimizers/sgd/. (accessed: 09.01.2021).

[54] *Computer Vision: Evolution and Promise*. 1996.

[55] Ojala T., Pietikäinen M., and Harwood D. "A Comparative Study of Texture Measures with Classification Based on Feature Distributions". In: 29 (1996), pp. 51–59.

[56] *Tensorflow Keras Module*. URL: https://www.tensorflow.org/api_docs/python/tf/keras. (accessed: 03.10.2021).

[57] P. Exarchos Themis, Papadopoulos Athanasios, and I. Fotiadis Dimitrios. "Handbook of Research on Advanced Techniques in Diagnostic Imaging and Biomedical Applications". In: 1st ed. Medical Information Science Reference, Mar. 2009. Chap. 22.

[58] Giannakopoulos Theodoros and Pikrakis Aggelos. "Introduction to Audio Analysis". In: (2014), pp. 59–103.

[59] *Violent Scenes Dataset*. URL: https://www.interdigital.com/data_sets/violent-scenes-dataset. (accessed: 25.07.2020).

[60] Xindong Wu et al. "Top 10 Algorithms in Data Mining". In: *Knowl. Inf. Syst.* 14.1 (Dec. 2007), pp. 1–37. ISSN: 0219-1377. DOI: 10.1007/s10115-007-0114-2. URL: https://doi.org/10.1007/s10115-007-0114-2.

[61] Min Xu et al. “Advances in Multimedia Information Processing”. In: *Springer Berlin Heidelberg* (2004), pp. 566–574.

[62] Yu Zhang and Qiang Yang. “A Survey on Multi-Task Learning”. In: *arXiv e-prints* (2017).