National University of Computer and Emerging Sciences, Lahore

VideoGuard: AI Based Video Censoring

Saif Ur Rehman 20L-1291 BSDS

Saadan Atif 20L-1360 BSDS

M.Raza Mehdi 20L-1294 BSDS

Supervisor: Dr. Aamir Wali

Final Year Project

June 6, 2024

Anti-Plagiarism Declaration

This is to declare that the above publication was produced under the:

Title: VideoGuard: AI Based Video Censoring

is the sole contribution of the author(s), and no part hereof has been reproduced as it is the basis (cut and paste) that can be considered Plagiarism. All referenced parts have been used to argue the idea and cited properly. I/We will be responsible and liable for any consequence if a violation of this declaration is determined.

Date: .....

Name: .....

Signature: .....

Name: .....

Signature: .....

Name: .....

Signature: .....

Author's Declaration

This states Authors' declaration that the work presented in the report is their own, and has not been submitted/presented previously to any other institution or organization.

Abstract

In our project, our main objective is to train an Artificial Intelligence model capable of identifying explicit scenes, violence, and offensive language within videos. To ensure the practicality of this trained model, we aim to create a user-friendly interface that allows users to play videos and apply various filters to remove different types of inappropriate scenes. Our ultimate project goal is to provide users with a viewing experience that is completely free of objectionable content.

Executive Summary

In today's digital age, objectionable content is increasingly prevalent in videos. Traditional manual filtering methods, as employed in the past, have become nearly impractical due to the sheer volume of content. Several companies have developed content moderation APIs for application use, but these often restrict end-user control. In contrast, VideoGuard empowers users with complete autonomy over content filtering. Users can specify the type of content they wish to filter, determine whether to remove or blur scenes, and decide whether to view content in real-time or save filtered videos for later use. These comprehensive features grant users an unprecedented level of control over content, a degree of control not offered by other solutions.

First and foremost, the project aims to engage in thorough research, designing, and implementing cutting-edge machine learning algorithms and advanced computer vision techniques to achieve highly accurate detection of offensive content within videos. Additionally, there is a strong emphasis on efficiency, with a goal to significantly reduce the time required for the removal of explicit material. The project also includes the categorization of explicit content into distinct types, such as nudity, violence, and foul language, to enable more precise and fine-grained content filtering. Ensuring compatibility with various video formats is a key priority to facilitate seamless use across different platforms. An intuitive and user-friendly interface (UI) has been developed, allowing users to easily play videos and providing customizable options for tailoring content filtering to individual preferences. Users have the capability to select videos from their computer directory, apply different content filters, specify level of filtering, and choose between cutting or blurring objectionable content. Ultimately, the tool also allow users to save censored videos for later use, providing a comprehensive solution for content moderation and enhancing the digital viewing experience.

The algorithms under use branch into three main categories: neural network architectures, modified deep learning models and supervised machine learning models.

To begin with, neural network architectures play a foundational role in various machine learning applications. The Artificial Neural Network (ANN) is a fundamental model consisting of interconnected layers of artificial neurons. Recurrent Neural Networks (RNNs) are tailored for sequential data, utilizing feedback loops that enable the retention of information. This property makes RNNs particularly suitable for speech recognition and time series analysis. Convolutional Neural Networks (CNNs), designed with specialized convolutional layers, have revolutionized image processing. They efficiently extract and recognize features in images, making them indispensable in computer vision applications.

Moreover, deep learning models represent the forefront of neural network architectures, each fine-tuned for specific tasks. ResNet50_V2, a variant of the ResNet architecture, is renowned for its deep residual

blocks, which facilitate the training of exceptionally deep networks. This architecture finds extensive use in computer vision tasks such as image classification and object detection. Inception_V3 is another standout model featuring inception modules that capture intricate patterns in images, making it invaluable for image classification and object detection. Meanwhile, DeepLab, a semantic segmentation model, harnesses dilated convolutions to capture fine-grained image details. This capability is crucial in tasks that require precise segmentation, such as autonomous driving and medical image analysis.

Ultimately, Machine Learning Algorithms constitute the bedrock of supervised learning, serving diverse problem domains. The Support Vector Machine (SVM) is a robust algorithm employed for both classification and regression tasks. Its strength lies in finding optimal decision boundaries by maximizing the margin between different classes. In contrast, Linear Regression, while simple, remains a foundational algorithm for predicting continuous target variables based on input features.

For our tool, we have selected multiple models to achieve the best results. Different versions of RNN, including GRU and LSTM, have been used to classify videos. Various CNN architectures have been employed to detect blood and nudity in a single frame of the video. A pre-trained speech-to-text model has been utilized to convert speech to text, and then using NLP, we detected foul words within a video. Utilizing different models allowed VideoGuard to detect violence, nudity, and profanity accurately.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

Throughout history, societies have repeatedly contended with the challenge of managing and overseeing content, particularly concerning explicit or inappropriate material's impact on their populace. This historical struggle can be observed in numerous instances, such as the ancient Roman Senate's pronouncements against indecent performances, the Roman Catholic Church's Index Librorum Prohibitorum that prohibited certain books, and the enforcement of the Motion Picture Production Code in Hollywood. These efforts to censor and control content have manifested in diverse ways across different eras. With the emergence of radio and television, fresh challenges arose, prompting governments and institutions to institute broadcasting standards and content rating systems. These initiatives were designed to safeguard vulnerable audiences, particularly children, from potentially harmful content.

In the digital age, the ascent of the internet and social media has brought forth unprecedented challenges and opportunities in the realm of content moderation. Online platforms have introduced intricate algorithms and content filtering systems to combat explicit material, hate speech, and misinformation, although their effectiveness has varied.

In today's digital era, objectionable content is becoming increasingly prevalent in videos. Manual filtering, as practiced in the past, has become nearly impossible due to the sheer volume of content. Various companies have developed content moderation **APIs** for use in applications, but these often limit end-user control. VideoGuard, on the other hand, empowers users with complete control over content filtering. Users can specify the type of content they want to filter, set age restrictions for group viewing, decide whether to remove or blur scenes, and choose whether to watch content in real-time or save filtered video for later use. These comprehensive features provide users with unprecedented control over content, a level of control not offered by other solutions.

This report will be structured into ten chapters. The initial chapter serves as an introduction, where we establish the project's purpose and identify its intended audience. Additionally, this chapter includes definitions, acronyms, and abbreviations that will be utilized throughout the report. Moving on to the second chapter, we delve into the vision for our project. Here, we clarify and expand upon the problem statement within its domain. We define our goals and objectives, including alignment with sustainable development goals. Moving on to Chapter Three, concentrates on our literature review, offering an overview of the research papers we've examined during the course of our project. Chapter four is about software requirements specifications. It mainly includes functional and non-functional requirements, use cases, and graphical user interface. Chapter five is about the methodology used in making VideoGuard. Chapter six includes the high-level and low-level diagrams of the project. Chapter seven is mainly related to implementation and test cases; it includes the training and testing of the models used. Chapter eight includes the user manual. Chapter nine is related to experiments conducted, and finally,

there is a conclusion chapter that concludes the report and presents future work.

1.1 Purpose of this Document

The purpose of this document is to introduce and elaborate on our project, which revolves around the creation of an innovative video filtering tool called VideoGuard. This tool is crafted to offer users a comprehensive solution for automatically detecting and removing explicit content from videos. Our primary focus is on crafting a simple **UI** that ensures a smooth and enjoyable user experience. At the core of our project lies the following research question: Can we develop a robust Windows based tool that enhances content moderation and user control in video consumption? This report will delve into our methodology, design, implementation, testing, and evaluation of the tool, while also delineating its limitations and potential pathways for future development.

1.2 Intended Audience

Our **FYP** report on the creation of a video filtering tool is intended for various important audiences. To begin with, our academic mentors and assessors hold a pivotal position in evaluating the excellence of our project. They will gauge our grasp of the subject matter, scrutinize the thoroughness of our research and development process, and assess the overall quality of our work.
In addition, researchers and academics engaged in studies related to content filtering and machine learning may also perceive our report as a valuable asset. It could serve as a source of inspiration and insight for their own research endeavors or as a reference point for future investigations and advancements in these fields.

1.3 Definitions, Acronyms, and Abbreviations

SDG: Sustainable Development Goal

FYP: Final Year Project

UI: User Interface

API: Application Programming Interface

CNN: Convolutional Neural Network

LR Learning Rate

ML: Machine Learning

PCA: Principle Component Analysis

VGG: Visual Geometry Group

MFCC: Mel-Frequency Cepstral Coefficient

SVM: Support Vector Machine

ANN: Artificial Neural Network

VSD: Voilence Scene Detection

DCT: Discrete Cosine Transform

RNN: Recurrent Neural Network

GRU: Gated Recurrent Unit

LSTM: Long Short-Term Memory

CV: Computer Vision

NLP: Natural Language Processing

1.4 Conclusion

In this chapter, we have undertaken a comprehensive exploration of our project, VideoGuard, a video filtering tool designed to automatically detect and remove explicit content. Our primary objective has been to provide users with a seamless and enjoyable experience through a user-friendly Windows-based tool. Furthermore, our report is intended to be a valuable resource for researchers and academics engaged in content filtering and machine learning studies. It stands as a source of inspiration and insight for their own research endeavors and future investigations in these fields. Despite acknowledging certain limitations, such as the need for continuous updates to adapt to evolving content, we remain optimistic about the potential pathways for future development. In essence, this chapter encapsulates our journey from inception to the creation of VideoGuard, emphasizing both its current contributions and its promising trajectory in the realm of innovative content moderation tools.

Chapter 2 Project Vision

The project envisions the development of an innovative content filtering tool aimed at enhancing user control over video content by automatically identifying and selectively removing explicit scenes, including nudity, violence, and foul language. Our vision is to provide a user-friendly, ethically designed solution that empowers individuals to tailor their media consumption while respecting legal boundaries, fostering a safer and more customizable viewing experience for a diverse range of audiences.

2.1 Problem Domain Overview

The VideoGuard will automatically analyze video content and selectively remove explicit scenes, including nudity, violence, and foul language. It will use CV, ML and NLP to identify and classify explicit content, followed by video and audio splitting techniques to either remove or censor those segments. Users will be able customize the level of filtering and choose from different methods of handling explicit content, offering a tailored viewing experience.

2.2 Problem Statement

The issue at hand centers on the proliferation of explicit content, including nudity, violence, and offensive language, in video content. It has become increasingly challenging to watch video content in a group that includes both children and teenagers. However, with the use of VideoGuard, such a group can enjoy a movie without concerns about inappropriate content that is not suitable for kids and teenagers.

2.3 Problem Elaboration

In this paper, we will follow a structured approach to develop VideoGuard, a tool for detecting objectionable content in videos and providing users with the means to filter it:

2.3.1 Content Detection Algorithms

We'll explore various **AI** algorithms to identify objectionable content within videos. This section will discuss the different techniques and their pros and cons.

2.3.2 Algorithm Selection

After evaluating the algorithms, we'll choose the most suitable one for VideoGuard based on factors like accuracy and efficiency.

2.3.3 Dataset Selection

To train and test our chosen algorithm, we'll need an appropriate dataset consisting of video and images containing objectionable content. We'll discuss the dataset collection and selection process.

2.3.4 Algorithm Implementation and Training

This section will detail the implementation of the selected algorithm and the training process using the chosen dataset. We'll describe how the algorithm learns to identify objectionable content.

2.3.5 User Interface Development

VideoGuard's user interface will be a crucial component. We'll discuss how it will function as a video player with filtering options. Users will be able to select a video to play, and the tool will detect and offer the option to remove objectionable content.

2.4 Goals and Objectives

We hope to achieve the following goals and objectives in our research and developement project.

- Research, design, and implement state-of-the-art machine learning algorithms and computer vision techniques for accurate detection of offensive content.
- To reduce the amount of time it takes to remove explicit content.
- Categorize explicit content into different types (e.g., nudity, violence, foul language) for more fine-grained filtering.
- Ensure that the tool is compatible with various video formats.
- Develop an intuitive and user-friendly interface **(UI)** allowing users to play videos.
- Implement customizable options for users to specify the extent of content filtering.
- Optimizing VideoGuard for real-time video processing to enhance its capabilities for future use.
- Expand the tool's usability by making it compatible with various streaming platforms.

2.5 Project Scope

The scope of this project is to develop a tool named VideoGuard, for filtering nudity, violence, and foul language from a video clip or a movie. VideoGuard will allow users to:

- Play a video by selecting it from their computer directory.
- Apply different content filters that they don't want in the video.
- Apply level of filtering to filter the content accordingly.
- Either cut the content or just blur it.
- Save the censored video for later use.

Initially the VideoGuard will be a Windows application, with simple user interface. But in future it will be expanded to various patforms, like Google Chrome extension. Also the work will be done in making the VideoGuard a real time censoring tool.

2.6 Sustainable Development Goal (SDG)

In this project we will be targeting SGD-3 (Good health and Well-being) and SGD-16 (Peace, Justice, and Strong Institutions):

- **SDG 3:** By protecting children from exposure to explicit content, our tool can help to improve their mental and emotional health.
- **SDG 16:** By helping platforms to maintain content standards, our tool can contribute to a more peaceful and just society.

Chapter 3 Literature Review / Related Work

3.1 Definitions, Acronyms, and Abbreviations

CNN: Convolutional Neural Network

LR Learning Rate

ML: Machine Learning

PCA: Principle Component Analysis

VGG: CNN architecture developed by Visual Geometry Group

MFCC: Mel-Frequency Cepstral Coefficient

SVM: Support Vector Machine

ANN: Artificial Neural Network

VSD: Voilence Scene Detection

DCT: Discrete Cosine Transform

RNN: Recurrent Neural Network

LSTM: Long Short-Term Memory

GRU: Gated Recurrent Unit

SLD: Structured Learning with Discriminative Models

CRFs: Conditional Random Fields

NLP: Natural Language Processing

ResNet_V2: Residual Networks Version 2

tanh: Hyperbolic Tangent

ReLU: Rectified Linear Unit

AFCR: Automated Film Censorship and Rating

HTML: Hyper Text Markup Language

CSS: Cascade Style Sheet

JS: JavaScript

3.2 Detailed Literature Review

Video censoring refers to the act of eliminating or concealing potentially offensive or inappropriate material within videos. This procedure serves various purposes, including shielding children from harmful content, adhering to governmental guidelines, or preventing the discomfort of viewers. In the contemporary age of exponential video content growth, manually overseeing and censoring objectionable

elements poses considerable challenges. In this section, we delve into a range of research endeavors aimed at constructing machine learning models capable of identifying sensitive content within videos.

3.2.1 Artificial Neural Network (ANN)

ANNs are computational models inspired by the structure and function of biological neurons. **ANNs** consist of layers of interconnected artificial neurons (perceptrons) that process data. Each connection between neurons has an associated weight, which is adjusted during training to optimize the network's performance.

The core operation in **ANNs** is the weighted sum of inputs, followed by the application of an activation function. Activation functions introduce non-linearity into the network, allowing it to approximate complex functions. Common activation functions include sigmoid, **tanh**, and **ReLU**.

Training **ANNs** involves forward and backward passes. The forward pass computes predictions, and the backward pass uses gradient descent optimization to update weights, minimizing a specified loss function [1]. Deep neural networks, including **CNNs** and **RNNs**, are specialized forms of **ANNs** tailored for specific tasks.

ANNs are versatile and applicable in various domains, including computer vision, speech recognition [2], and **NLP**. In computer vision, **CNNs**, a type of **ANN**, excel in image classification and object detection. In **NLP**, **ANNs** are used for text classification, sentiment analysis, and machine translation.

3.2.2 Recurrent Neural Network (RNN)

RNNs are specialized neural networks designed for processing sequential data. Unlike feedforward neural networks, **RNNs** possess feedback connections that allow them to maintain hidden states and capture temporal dependencies. Each unit in an **RNN** processes input data while incorporating information from previous time steps, making it suitable for sequences of arbitrary lengths. The core mathematical operation of an **RNN** unit involves computing the hidden state at each time step using a combination of the current input and the previous hidden state. This hidden state carries contextual information about prior inputs, enabling the network to model sequential dependencies.

However, standard **RNNs** have limitations in handling long-range dependencies and suffer from the vanishing gradient problem, which hinders the learning of long sequences. Variants like **LSTM** and **GRU** were introduced to address these issues [3]. **RNNs** find application in tasks requiring sequence modeling, including speech recognition [4], language modeling, and time series prediction. In **NLP**, they excel in tasks like machine translation, text summarization, and sentiment analysis, where context and order of words are vital.

3.2.3 Structured Learning with Discriminative Models (SLD)

SLD is a framework tailored for structured output prediction tasks, where the output exhibits complex relationships and structures. **SLD** emphasizes the direct modeling of dependencies between input and output variables, as opposed to considering them separately [5]. In **SLD**, structured output prediction is formulated as an optimization problem, often employing structured prediction techniques like **CRFs** or structured **SVMs**. These models explicitly encode relationships between output labels and ensure that predictions conform to global dependencies and constraints.

SLD is applied in various fields, including computer vision and **NLP** [6]. In computer vision, it is valuable for tasks such as image segmentation, where pixel-level labels are interdependent. In **NLP**, **SLD** is used for named entity recognition, where recognizing entities correctly requires considering the context and relationships between words.

3.2.4 ResNet_V2

ResNet_V2 addresses the vanishing gradient problem in deep neural networks. Traditional deep networks suffer from difficulties in training when they become very deep, as gradients tend to vanish during backpropagation, hindering weight updates.

ResNet_V2 introduces residual connections, also known as skip connections or shortcuts, between layers. These connections allow the gradient to flow more efficiently during training. Instead of learning the desired mapping directly, **ResNet_V2** learns residual functions. The output of a layer is the sum of its input and the learned residual function, which facilitates the propagation of gradients through the network [7]. This architectural innovation enables the training of very deep neural networks with hundreds of layers, improving performance without degradation.

ResNet_V2 is predominantly applied in computer vision tasks, such as image classification and object detection, where deep networks are required to achieve state-of-the-art performance. It serves as a foundational architecture for various deep learning applications in computer vision [7].

3.2.5 CNN and InceptionV3

Convolutional Neural Networks (CNNs) have brought about a profound transformation in the realm of computer vision and image processing. CNNs constitute a category of deep neural networks meticulously engineered to autonomously and flexibly acquire spatial hierarchies of features from input images. CNNs have discovered utility in numerous domains, encompassing image categorization, object identification, facial authentication, and the analysis of medical images.

3.2.5.1 Inception architectures

The Inception architecture, which is alternatively recognized as GoogleNet, constitutes a series of CNN architectures devised by Google's research team. Inception networks have garnered recognition for their effectiveness in terms of computational resources, concurrently attaining top-tier performance in image classification assignments. InceptionV3 stands out as one of the most prominent models within this family.

3.2.5.2 InceptionV3 Overview

InceptionV3 made its debut in the paper [8]. This architectural innovation was conceived to tackle the intricacies associated with constructing deep networks containing an extensive array of parameters, all the while ensuring that computational complexity remains manageable. Key characteristics and components of InceptionV3 include:

- **Inception Modules:** InceptionV3 incorporates a component known as the "Inception module" or "Google Inception module." These modules leverage multiple parallel convolutional filters with distinct sizes and incorporate pooling operations to seize features at diverse spatial scales. This architectural choice empowers the network to encompass both detailed and broad-spectrum information from the input data.
- **Factorization:** In order to diminish computational complexity, InceptionV3 employs factorized convolutions, including 1x1 convolutions, renowned for their computational efficiency. These 1x1 convolutions play a pivotal role in reducing the depth of feature maps and, consequently, curtailing the overall count of parameters within the model.
- **Auxiliary Classifiers:** InceptionV3 further incorporates auxiliary classifiers at intermediate layers of the network, a concept elucidated in the reference [8]. These supplementary classifiers were introduced as a means to combat the vanishing gradient predicament and facilitate the training of more profound networks.

3.2.5.3 Performance and Applications

InceptionV3 has shown outstanding performance on different standard datasets, like VSD [9]. Its clever design has made it a go-to option for practical uses, such as rating movies based on their content [10].

3.2.6 SVM

SVM is a supervised machine learning algorithm that can be used for both classification and regression tasks. SVM is particularly suited for high-dimensional and non-linearly separable data. **SVM** works

by finding a hyperplane in the input space that optimally partitions the training data into two classes [11]. The hyperplane is chosen to maximize the distance between the two classes. This is the distance between the hyperplane and the nearest training point for each class [11].

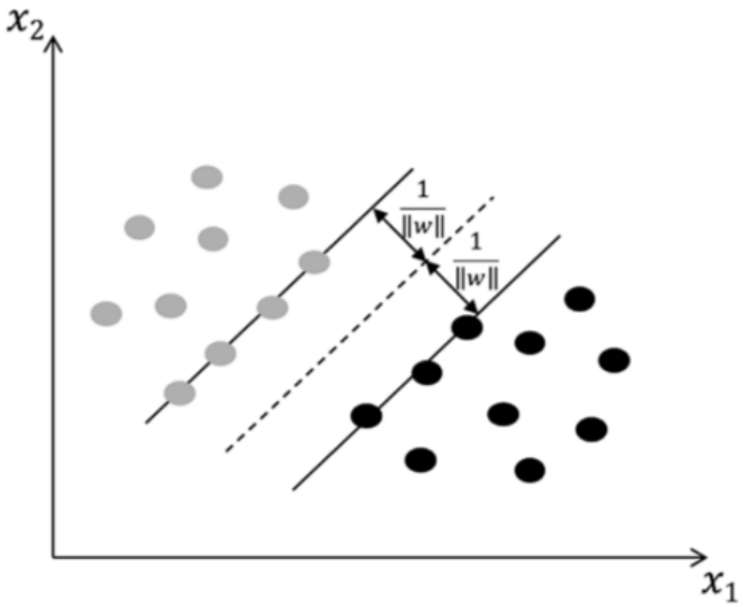

Figure 3.1: Support Vector Machine

an image taken from the article [11]

Once an **SVM** is trained, it can be used to classify new data points by predicting which side of the hyperplane they fall on. **SVM** can also be used to perform regression by predicting a continuous value for new data points, such as the price of a house or the risk of a patient developing a disease [11].
SVM is a popular algorithm and has been used in many different types of classification. Upon identifying a scene in a video that might be objectionable, the SVM has been trained on movie-level features to classify the movie plot into one of three types of violent content, e.g., low, medium and high [12]. A similar strategy is used by paper [13], in which **CNN** is used to extract the features and then SVM is used for binary classification.

3.2.7 3D Convolutional Neural Network

In the realm of content detection, 3D CNNs find application. A specific architecture known as a two-stream 3D CNN can be employed for analyzing videos containing various activities, as discussed in [14]. This architecture comprises two separate streams: one stream focuses on extracting spatial features, while the other stream concentrates on extracting temporal features. These two streams work together, combining their insights, to make a final prediction about the content of the video.

- **Spatial features:** They refer to the details about the stationary or unchanging aspects of data. In the context of images or videos, spatial features provide information about what objects or

elements are present within a frame, where they are positioned, their shapes, sizes, colors, and various other characteristics. Essentially, these features capture the static or non-temporal aspects of the visual content, helping to understand what's in the image or video and where it is.

- **Temporal features:** They pertain to the data's dynamic aspects, focusing on how objects or elements evolve, shift, or engage with one another across a sequence of frames or time intervals. In the context of videos or time-series data, these features describe the transformations, movements, or interactions that occur over time, helping to capture the changing nature of the data as it unfolds.

This approach has undergone evaluation on various publicly available violence detection datasets, including the RWF-2000 dataset [15], the Violence Dataset SCVD [16], and the ViolentFlows dataset [17]. While this technique shows significant advancement, it's important to note that it is still in the process of development and may continue to evolve. Here we mentioned some challenges in using 3D convolutional networks:

- One of the major hurdles when applying 3D CNNs for violence detection is the need for extensive training data. This demand arises because 3D CNNs possess a substantial number of parameters and must acquire knowledge about both spatial and temporal features from the data. In order to train effectively, these networks require a substantial volume of examples to capture the nuances of violent behavior accurately.
- Another obstacle in the use of 3D CNNs for violence detection is the computational expense involved in both training and deployment. This arises because 3D CNNs have to handle video data, which is significantly more computationally demanding than image data

Instead of relying solely on 3D convolutional networks, which have their drawbacks, using multiple machine learning models to analyze both video and audio data, as discussed in [18], can be highly effective. In scenarios like violent scenes, there often are audio cues like screaming or loud noises that can provide crucial context. By harnessing audio data alongside visual information, violence detection can be enhanced, potentially leading to more accurate results. This multimodal approach leverages the strengths of both video and audio analysis for a more comprehensive understanding of complex scenes.

3.2.8 MFCCs

MFCCs, a commonly employed method in the realm of speech and audio processing, serve as a means of feature extraction. They stem from the mel-frequency cepstrum, which represents the power spectrum of sound over a brief time interval. This representation is constructed by applying a linear cosine transformation to a logarithmic power spectrum situated on a non-linear mel frequency scale. MFCCs are typically extracted from an audio signal using the following steps:

- The audio signal is partitioned into brief frames, usually spanning 20-30 milliseconds each.
- For each frame, a Fourier transform is employed to compute the power spectrum.
- The power spectrum is then transformed into the mel scale, which is a frequency scale that non-linearly mimics human auditory frequency perception.
- Subsequently, the logarithm of the power spectrum in the mel scale is calculated.
- Finally, the **DCT** is applied to the logarithmic mel-scaled power spectrum to yield the MFCCs.

The initial **MFCC** coefficients typically encapsulate the primary details regarding the spectral characteristics of the sound signal, focusing on the broader spectral envelope. Conversely, the higher-order coefficients delve into finer, more intricate spectral information. Leveraging MFCCs in conjunction with video content can enhance the accuracy and effectiveness of content filtering tools like VideoGuard. Integrating these coefficients into the system can lead to a more precise and robust content filtering solution [19].

3.2.9 AudioVGG

AudioVGG is a deep learning model used for audio embedding [20]. It has been applied in various fields including content moderation, video classification, multimodal learning, and movie content classification. In the context of content moderation and movie content ratings, AudioVGG has been used to automate the video content rating process [18]. This is especially useful as the amount of online video content increases, making manual review by committees increasingly difficult. The use of AudioVGG in these areas is part of a broader trend toward AFCR, which is becoming an important application of ML. AFCR aims to generate automated, aggregated ratings for movies or any media to determine the appropriate minimum age for viewing the content.
AudioVGG is based on the principles of the **VGG** architecture, a classical convolutional neural network [20]. The **VGG** network uses small 3x3 filters and is characterized by its simplicity, with the only other components being pooling layers and a fully connected layer. In the case of AudioVGG, it works by extracting short audio clips from videos [20]. These audio clips are then processed over the network to create audio embeddings [18]. These integrations can be used for a variety of applications such as content moderation, video classification, multimodal learning, and movie content ranking [18].

3.2.10 Image based Linear Regression

Linear regression is a predictive statistical model used to model the relationship between a dependent variable and one or more independent variables. Linear regression aims to find the line of best fit. It assumes that the relation is linear, errors are not correlated, homoscedasticity, and normally distributed

error [21].

3.2.10.1 Visual Dataset

Dataset used for training the model is divided into training and testing datasets. Training dataset consists of almost 70-80% of the whole data and the rest is the testing data [21]. As the model aims to detect objectionable data in movies and assign those images specific vulgarity factors, we need a dataset of images to detect the objectionable objects in videos and to detect abusive language we require a textual dataset. We will resize all the images in the dataset to the same size before inputting these images to our model, we will resize the images to 128 x 128. Since the dataset of images is made up of pixel data so there is no need for an imputer strategy to find missing values [22]. The multiple linear regression equation:

$$y = a0 + a1x1 + a2x2 + a3x3 + \ldots + anxn$$

Where n is the number of independent variables in the equation, a0,a1,a2..., are the predictor coefficients and ×0, ×1, ×2, ... xn are the independent variables and y is the dependent variable. In this case, n is the total number of pixels in the image. Based on the values of dependent variable and independent variables, extracted from the dataset, the aim is to find best coefficients that will fit the linear equation to predict the new videos [22]. It is not compulsory that our model use all the coefficients as using all the coefficients may result in poor accuracy. Methods to decide which coefficients will be needed:

- **All in:** All the coefficients are used ignoring the p values.
- **Backward elimination:** A threshold level is set to determine which coefficients will be kept and which coefficients will be discarded. Higher P value means the variable will produce more inaccurate results.
- **Forward selection:** We start with an empty model and iteratively add variables to it. If at any stage the P value of the model exceeds the threshold value we will diacard the current model and use the precious one.
- **Bidirectional elimination:** We perform both backward elimination and forward selection simultaneously. This process continues till entering or removing any other variable does not make much difference in P value of the model.

As inappropriate visual content is not always appearing in a specific part of image but it can be located anywhere in the image so the best option is to use All in techniques so all the image parts can be determined. The pixel values are the linearly independent variables and the level or severity of objectionable content is a linearly dependent variable [22].

3.2.10.2 Textual Dataset

Training a model for the detection of inappropriate auditory content, a textual dataset containing a large number of commonly used English words can be used. Since the model only takes numerical values as input, we first need to convert string data to numerical. For this, label encoding will be used to assign a unique number to every word in the dataset. However, this creates a problem. For instance, if the word ”hi” is encoded to 1 and the word ”hello” is encoded to 2, ”hello” has a higher value than ”hi”, potentially impacting the calculation of parameters of the model. One solution is to use one-hot encoding, which will create a separate column for each unique word in the dataset, making it an independent variable. Each column will have a value of 1 where the words match with words in the dataset before encoding and 0 where they do not match [23]. Using dummy variables presents another problem known as the dummy trap. Since all variables are columns with values of 1s and 0s, there is a high chance of dependence or collinearity between the independent variables, which can confuse the regression model. To avoid this issue, one variable can be omitted, establishing it as a reference point for comparison [22].

3.2.10.3 Processing of Image Dataset

A single image or frame captured from a movie can be represented in a 3d array. The three dimensions are an image in the row, column, and the pixel index in that particular row and column. Shape of an image is very important for an image recognition system as it needs to compare the image with images it has already been fed with. For detection of objects in the image we will convert the image to a black and white image. The benefit of a black and white image is to simplify the data as converting to black and white makes it easier to focus on the structural appeal of the image. Colored images may contain noise in the color channels, which can interfere with image analysis. By using black and white images, the influence of color noise is greatly reduced, resulting in cleaner data. To convert to the black and white image we find the average pixel value and all the pixels below the average value are given 0 value representing black color and pixels equal or higher than the average value are given value 255 representing white color [22].

3.2.10.4 Detection of Inappropriate Images

These black and white images are fed to the model. The model deals each pixel value as an independent variable and the vulgarity factor of the detected object is the dependent variable. Now we will train the model using the training dataset to find and optimize the linear regression parameters. A threshold value will be set indicating that if an image vulgarity factor is above this threshold then it will be considered as inappropriate. Such a model is trained on pornographic images, sexual content and neutral images with no sexual or pornographic content.

3.2.10.5 Detection of Inappropriate Auditory Words

The Encoded dataset representing the words is given to the model for training purpose and set a threshold vulgarity factor to assign it the label of being appropriate or inappropriate.

Word	Vulgarity Factor
hi	0.1
hello	0,1
****	5
*****	6.1

Figure 3.2: Detection of Inappropriate Auditory Words

an image taken from the article [22]

3.2.10.6 Limitations in Implementing Linear Regression Model

The model assumes that dataset is linearly separable and for increasing the accuracy of the model images and words which have to be fed in form of the training phase have to be extremely large.

3.2.11 DeepLab

DeepLab is a convolutional neural network used in the field of computer vision for semantic image segmentation. It classifies the image pixels to identify objects. It has multiple variants and uses techniques like atrous convolution and spatial pyramid pooling to capture contextual information at different scales making it a powerful tool for the essential task of detecting objects in computer vision applications [24].

3.2.11.1 Detecting and Removing Profanity from Videos

Deeplab has been used to remove profanity from text. The aim of the model is to silence the audio and pixelate the lips if present in a video segment containing profanity. Although there are various methods to detect it, a profanity removal mechanism is also needed. A software can be developed which takes user video in input and silences the profanity-containing segment and pixelates the lips and output video will be completely profanity free [24].

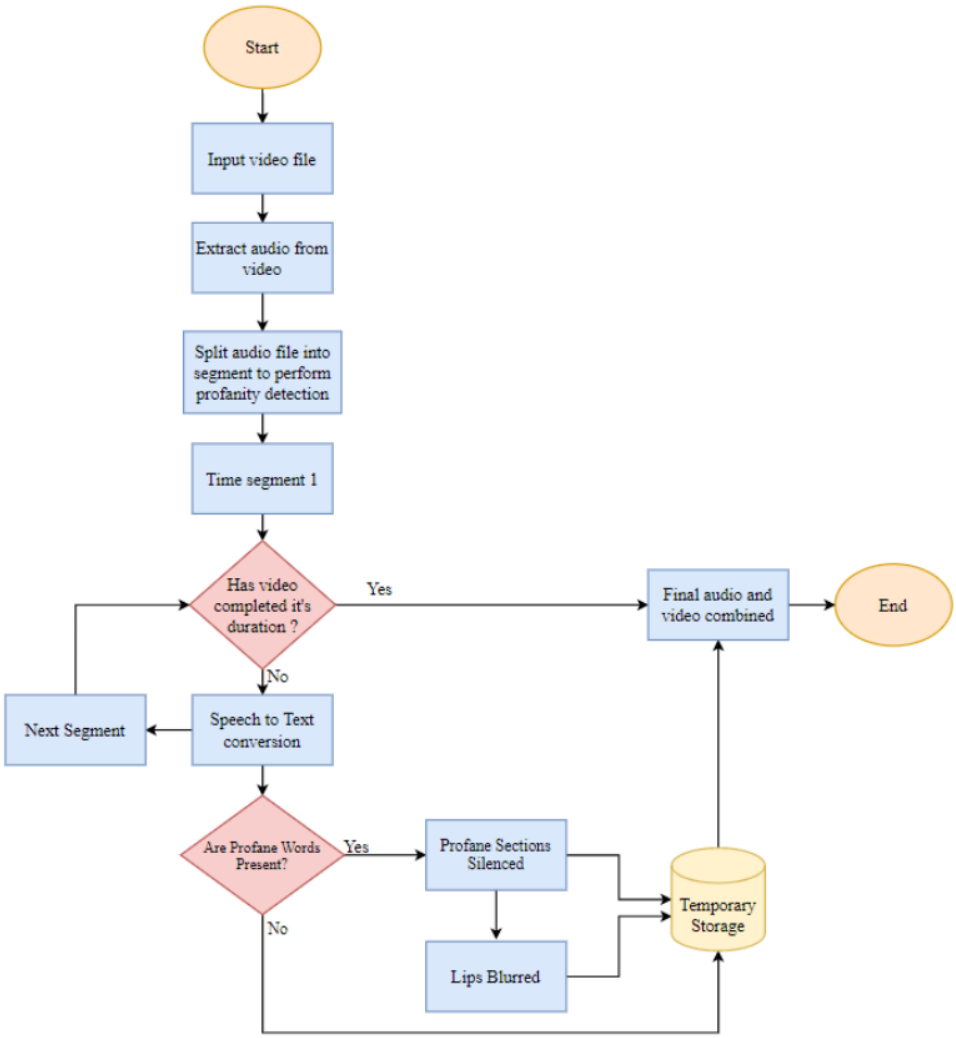

Figure 3.3: A Profanity Removing System

an image taken from the article [24]

The two major part of the systems are **silencing the audio** and **pixelating the lips**.

3.2.11.2 Silencing the Audio

The audio is first converted to text using speech to text library and then the text is checked for profane words by using a large dataset containing profane words. The audio is converted to text in time segments of 3 seconds each. If a 3 seconds segment contains profanity words that segment is further divided into 2 seconds segments and then 1 seconds segment to get the exact time at which the profane word was used. This time is recorded and later provided as input to the system to mute the audio at that time and pixelate the lips.

3.2.11.3 Pixelate the Lips

For this openCV is used to detect the lips in a frame. Facial landmarks are the points on the face that represent a particular part of the face. First we need to detect where the face is located in a frame and for

this we will use an openCV detector which is pre-trained by extracting Histograms of Gradients(HOG) from images followed by linear SVM. After getting the face locations, a facial landmark detector is used to get the coordinates of the lips and these coordinates are given to lip -blur function which will create a rectangle over the lip to blur it. The lip-blur function will take maximum and minimum values of x and y from the coordinates list passed by facial landmark detector and will find the width and height of the lip:

$$\text{Lip width} = \max_{x} - \min_{x}$$

$$\text{Lip height} = \max_{y} - \min_{y}$$

This way the rectangle area around the lip is found and blurred. For blurring we use a gaussian blur filter on each pixel of the detected region. This process is applied on all the extracted frames between the segments that were detected as containing profane words. These extracted frames after getting the lips blurred if any present are merged back with the rest of the video. Below is the example of lips blurring:

Figure 3.4: Input Frame

an image taken from the article [24]

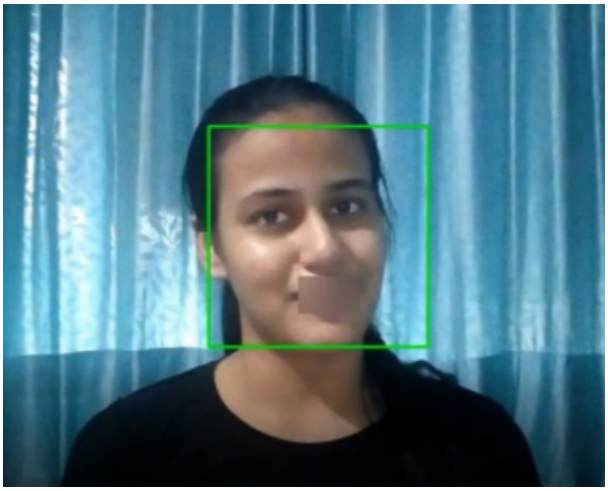

Figure 3.5: Blured Frame

an image taken from the article [24]

3.2.11.4 Result

The system was tested on 50 videos which contained 187 profane sections out of which 154 were detected and then removed.These videos collected from different sources like Facebook, Youtube etc. Accuracy of the system was calculated as follows:

Accuracy = (no of profane sections moderated / no of profane sections present) * 100

Accuracy = (154 / 187) * 100

Accuracy = 82.35% [24]

3.2.12 Text-To-Speech Model

For classifying the audio into vulgar or not, the audio is converted to text using a text-to-speech model, as mentioned above. Silero [25] is a pretrained model that can convert audio to text extremely fast on CPU with good accuracy. Here, accuracy does not matter a lot because the main goal is to detect a vulgar word in the text. Therefore, it does not matter whether the text makes complete sense or not; the main concern is whether the vulgar word in the audio is converted to text or not.

3.2.13 Related Research Work

3.2.13.1 Critical Analysis of the Research Item

CNN-based architectures have been widely used for classification of inappropriate content. From the work of Aldahoul et al. [26] it was observed that despite the high accuracy of CNN model on images, it had high rate of false positive and false negative on video data, which shows the inconsistency in the accuracy because the temporal nature of video dataset had not been considered. RNN-based models

or RNN versions including GRU and LSTM, which are relatively recent in their development, also assist in examining sequential characteristics and yield improved outcomes when integrated with other machine learning or deep learning techniques. As demonstrated in Daniel et al.'s study [19], adopting a multimodal approach can enhance the architecture's ability to assess the intensity of content in movies for implementing AFCR.

3.2.13.2 Relationship to the Proposed Research Work

To implement an automated video censoring system, we need to use multiple machine learning models. For blood detection, a CNN model can be used. Each video frame can be split into smaller sections, say 16 subsections, and then CNN can detect blood in each subsection. Similarly, CNN architecture can be used for detecting nudity, porn, and sexual content in a video frame, as in such types of images, we are only concerned about human skin and sexual organs. CNN can also be used to detect weapons within a video frame, which can help identify a violent scene. RNN versions like LSTM and GRU can be used to detect scenes in which temporal nature is crucial, such as fighting, gunshots, etc. We first need a diverse dataset consisting of different types of labeled video clips to train our video model. We will process the frames of the video by initially resizing all the frames to a specified standard size and converting them to black and white images before inputting them to our model. For the detection of profanity in textual data, we need to convert audio to text and then, using NLP, we can detect foul words. Our aim is to provide the benefit of machine learning models to detect inappropriate content in the form of a video player with an interactive UI to help the user apply different filters to censor different types of inappropriate content. RNN combined with other CNN models has proven to be the most accurate for the detection of inappropriate content, so this combination of models will play an important role in the development of our tool.7

3.3 Literature Review Summary Table

Table 3.1: Application Review Summary Table

This table contains some applications related to our project and their features, relevance and limitations.

Application	Features	Relevance to Application	Limitations
Google cloud video intelligence api.	Explicit content detection and moderation.and Video labeling, object tracking, shot change detection, speech recognition, and timestamping.	Content moderation for detecting explicit content and Video classification, object tracking, scene detection, and sentiment analysis	Potential cost considerations and accuracy may vary, especially in complex content
Microsoft video indexer	Automatic detection and moderation of explicit content And Video classification, speech transcription, object tracking, and sentiment analysis	Content moderation for explicit content. and Video categorization, metadata extraction, and object tracking	Potential costs associated with usage and accuracy variations in complex content, language support constraints, and content length limitations
Profanity Filters	Detects and filters offensive language and customizable, considers context, and can operate in real-time	Content moderation in online platforms and ensuring appropriate language in messaging apps, games, and parental controls	Challenges with multilingual content and cultural variations.

Table 3.2: Summary of Literature Review

This table contains the summary of research papers used in our literature review. Authors, method used, results and limitations of their work.

Author	Method	Results	Limitations
Su et al. [14]	3D CNN	Improved accuracy results by nearly 8% on the RWF-2000 dataset [15]	Computationally expensive
Freitas et al. [18]	InceptionV3, AudioVGG and SVM	Achieved F1-score of 98.95% for appropriate videos and 98.94% for inappropriate videos	Detects 5 seconds video segments only
Gruosso et al. [10]	CNN and InceptionV3	Achieved an accuracy of 90.9% for binary classification and 76.1% for multiple classification	High false discovery rate, did not test other content
Martinez et al. [12]	CNN	Achieved 60.9% precision and 60% recall	Sentiment related features are not used
Lyn et al. [13]	CNN and SVM	Achieved an accuracy of 92.80% and 93% precision	Hyperparameters did not tune practically and systematically, temporal nature of the dataset not utilised
Daniel et al. [19]	MFCCs	Achieved an audio accuracy of 80.98%	All sensitive contents are not addressed
Susanty et al. [2]	ANN	Achieved an accuracy of 96.8% on textual data	Cannot be used for detecting other content
Emon et al. [4]	RNN	Achieved an accuracy of 82.20% on textual data	Cannot be used for detecting other content
Fastest train and test data processing time. Han et al. [6]	SLD	Achieved an accuracy of 98.31% on adult content	Temporal nature of the dataset not utilised
Binary classification done on feature vectors extracted from various CNNs using SVM to detects inappropriate content with complex background. Aldahoul et al. [26]	CNN and ResNet_V2	Achieved an accuracy of 87.87%	Problem of small-scale regions inside the frames not considered, temporal nature of the dataset not utilised

3.4 Conclusion

In this section we discussed the potential of video censoring and its utility for end user leading up to methodologies we can use to achieve this , which are ANN, SLD, RNN, ResNet_V2, CNNs, Inception architectures, SVM, MFCC, Deeplab and and audio VGG. We discussed each of these techniques in detail with examples of already existing implementations and their uses.

Chapter 4 Software Requirement Specifications

This section will describe all modules of system requirements and design along with the necessary diagram and figures. It will describe functional requirements, design constraints, and other factors necessary to provide a complete and comprehensive description of the software.

4.1 List of Features

Our system will have the following list of features

- Video playback with censorship options.
- User can select censorship type (nudity, violence, profanity).
- Automatic detection and censorship of explicit/violent scenes using deep learning models.
- User can select weather to blur or cut any violent/explicit scene.
- Real-time muting of explicit words via audio analysis during playback.
- Users can add words they think are appropriate and they will be beeped or muted next time.
- Custom model training using labeled dataset to improve censorship accuracy over time.

4.2 Functional Requirements

Table 4.1: Functional Requirements

This table includes functional requirements of our project

ID	Description	Priority
FR1	Allow video playback in common formats like MP4, AVI	High
FR2	Prompt user to select censorship level before playback	High
FR3	Skip/mute/blur scenes in video dynamically based on censorship level	High
FR4	Provide settings to customize censorship filters	Medium
FR5	System must have a user friendly interface where users can easily use features like rewind, fast forward, play and pause	Medium

4.3 Quality Attributes/Non-Functional Requirements

Table 4.2: Non-Functional Requirements

This table includes none-functional requirements of our project

Attribute	Description	Target
Accuracy	Precision in detecting explicit content	90%
Latency	Delay between detection and censorship	1 sec
Usability	Configuration options for customization	Easy to use
Scalability	Efficient use of system resources (CPU, memory) to prevent excessive consumption during video playback.	Minimum
Compatibility	The video player should be compatible with various versions of the Windows operating system.	Windows 8 and above

4.4 Assumptions

The assumptions made for the specification are: videos have audio/visual contents, sufficient labeled data available for model training and private personal viewing with consent.

4.5 Use Cases

Table 4.3: Video Selection

This table provides the use case of video selection from computer directory.

Name	Video Selection		
Actors	User		
Summary	Allows user to upload video file for censorship.		
Pre-Conditions	Supported video formats, max duration.		
Post-Conditions	Video uploaded successfully for further processing.		
Special Requirements	None		
Basic Flow			
Actor Action		**System Response**	
1	The user clicks on the video selection button.	1	System opens file explorer to allow video file selection.
2	User selects video file from directory.	2	System opens the selected video and redirect to video player layout.
Alternative Flow			
3	File format not supported.	2-A	The system responds with an error message: Format not supported.

Table 4.4: Censorship Configuration

This table provides the use case of video selection from computer directory.

Name	Configure Censorship		
Actors	User		
Summary	The user selects censorship after the video has been processed.		
Pre-Conditions	Valid video file uploaded.		
Post-Conditions	Censorship settings (nudity, violence and profanity) applied.		
Special Requirements	None		
Basic Flow			
Actor Action		**System Response**	
1	System allows user to select censorship filters.	1	User chooses from nudity, violence, and profanity filters.
2	User selects save and return.	2	System saves selected censorship level and return to video playing layout.
Alternative Flow			
3	User does not select.	1-A	Previously selected settings applied.

Table 4.5: Video Playback and Censorship

This table provides the use case of video selection from computer directory.

Name	Video Playback		
Actors	User		
Summary	Video plays with selected censorship filters.		
Pre-Conditions	Valid video uploaded and censorship configured.		
Post-Conditions	Video played back with scenes skipped/muted based on censorship level.		
Special Requirements	None		
Basic Flow			
Actor Action		**System Response**	
1	User starts video playback.	1	System detects inappropriate scenes based on censorship level. Detected scenes are skipped, muted or blurred. Video playback continues with applied censorship.
Alternative Flow			
2	User fast forward the video.	1-A	System takes a slight pause, re-calculate initial processing and continue playback.

4.6 Hardware and Software Requirements

This section covers the hardware and software requirements that will be required to develop and deploy the project.

4.6.1 Hardware Requirements

- Sufficient GPU for training the model.

- User's machine must also have a dedicated GPU to boost video decoding tasks and improve overall performance.

4.6.2 Software Requirements

- Windows 8, 10 or 11.
- Python 3.11, TensorFlow 2.0, PyTorch, OpenCV.
- Other libraries such as Matplotlib, NumPy and Pandas.
- Electron JS, HTML, CSS and JavaScripts.
- Tools like Visual Studio Code, PyCharm, or Jupyter Notebooks.

4.7 Graphical User Interface

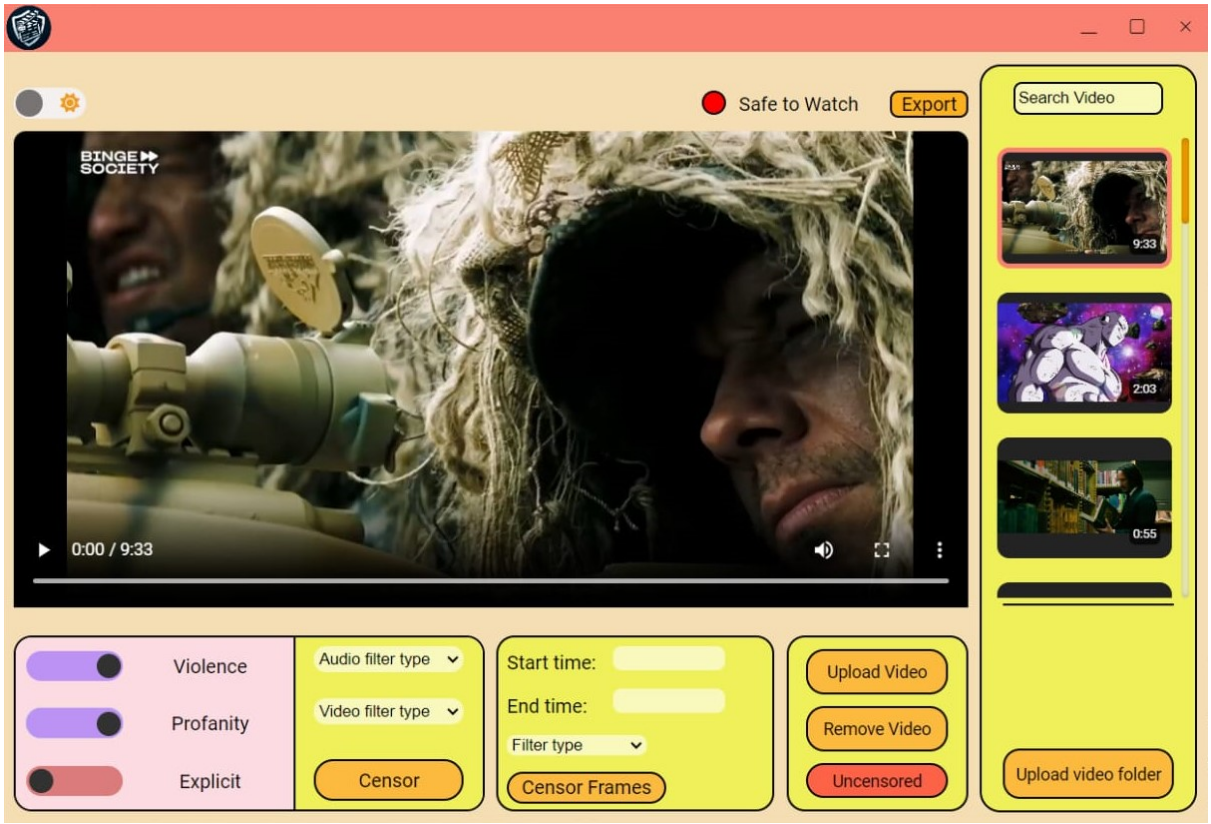

Figure 4.1: A light themed view of VideoGuard

The GUI has an option to upload video and the settings tab contains the filters and level of filtering the user will after the selected video has been processed.

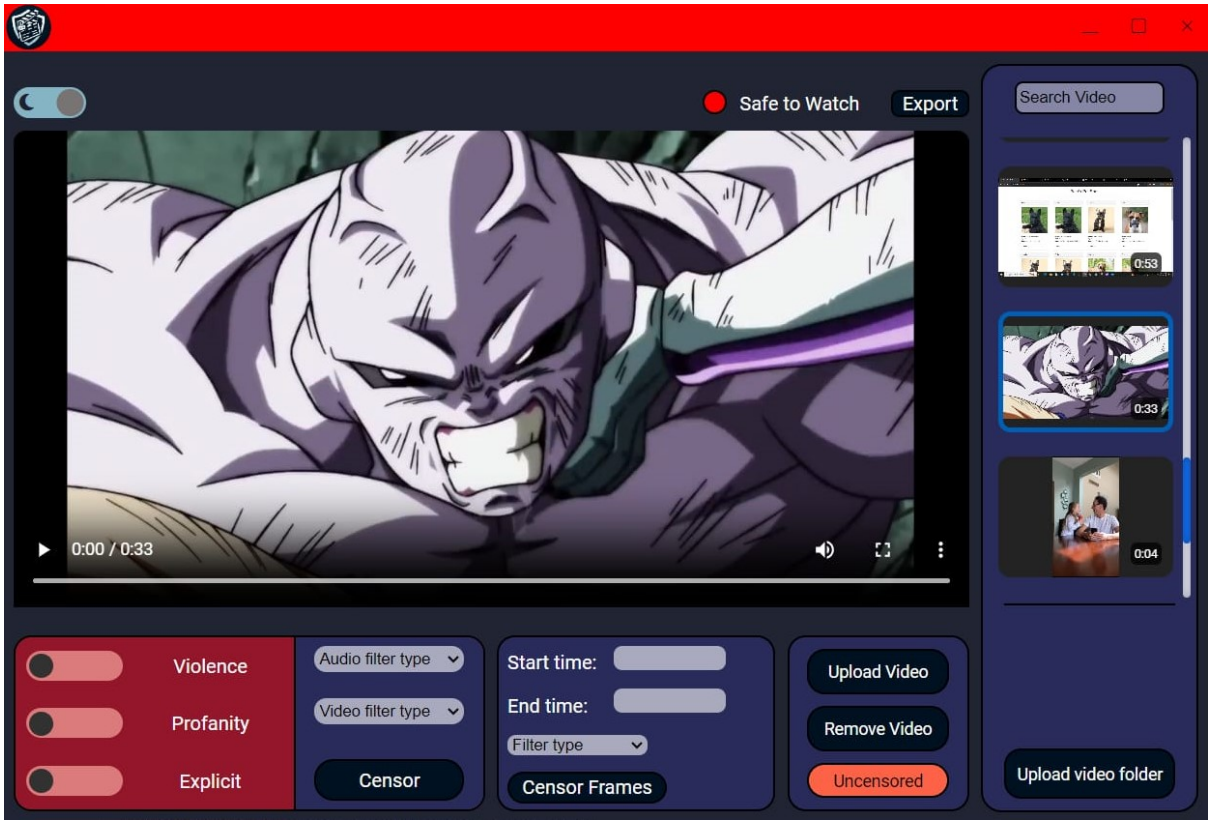

Figure 4.2: A dark themed view of VideoGuard

The GUI has an option to upload video and the settings tab contains the filters and level of filtering the user will after the selected video has been processed.

4.8 Risk Analysis

This section explain the risks that may be encountered during the project.

4.8.1 Code Issues

- Inefficient code.
- Logical errors.

4.8.2 Schedule Risk

- Inaccurate time estimation.
- Project scope expansion.
- Failure in completion.

4.8.3 Unmet Expectations

- Imprecise budgeting.

- Inaccurate outputs.
- Incorrect estimation of deadlines.

4.8.4 Budget Issues

- Overruns in costs.
- Unpredicted growth of project scope.
- Inadequate budget estimation.

4.8.5 Technical Risks

- Use of outdated technology which has no scope in future.
- Extremely complex implementations.
- Module integration issues.
- Excess censorship due to classification errors.

4.9 Conclusion

Overall, this SRS provides comprehensive details on Video Guard's scope, external interfaces, logical design - equipping the software development team with information required to architect and develop the system accordingly. Additional detail incorporation and validation across modules would further harden the foundations. With practical design and development best practices, the outlined solution can be brought to life effectively.

Chapter 5 Proposed Approach and Methodology

This chapter will discuss the proposed approach and methodology of our VideoGuard tool, explaining how it will filter out frames with explicit content and how it will remove foul words. It will delve into the details of how a user-selected video will be filtered in our tool.

5.1 Dataset And Model Training

5.1.1 Model Selection

We selected several models for filtering explicit frames and language from the video. As discussed in Chapter 3, RNN addresses the temporal nature of the video, thereby yielding better results than other models. We used RNN for scenes in which temporal nature is crucial, such as gunshots and stabbings. For nudity, sexual figures, and pornography, we have used a model trained on the NSFW dataset. This model is capable of detecting nudity, sexual dressing, and pornography within an image or, in our case, a video frame. For detecting profanity, we are using the Silero speech-to-text model, which converts audio into text with high accuracy, and after that, we detect foul words in the text. As detecting inappropriate content in a video is not easy, using several models is a way to achieve better performance.

5.1.2 Dataset Collection

Collecting data from various movies and TV series poses a challenging task throughout the entire journey. Initially, we collected movies or series containing numerous explicit scenes. The data collection process involves extracting both explicit and non-explicit scenes from each video file. Subsequently, we compile two video files for each original video: one combining all explicit scenes and the other combining all non-explicit scenes (though they may appear explicit). These two videos for each original file are then inputted to a Python-based splitter program. This program generates small clips of uniform length, such as 100 frames (also referred to as the window size, which will be discussed shortly). For instance, if the explicit video contains a total of 1000 frames, the splitter creates 10 small videos, each consisting of 100 frames. The same splitting process is applied to non-explicit videos.

5.1.3 Feature Extraction

After collecting the data, we extracted features from each small video. Following the addition of labels, we will have completed our two datasets: one for explicit scenes and one for non-explicit scenes. For feature extraction, we used ResNet50.

5.1.4 Model Training

After completing the challenging data collection phase, we trained different RNN versions, including GRU and LSTM, which are famous for solving the vanishing gradient problem. We split the data into 20% for testing and 80% for training. The models are provided with the extracted features along with their corresponding labels during the training phase.

5.2 Working With User Video

5.2.1 Taking User Input

The user will open the media player on a Windows PC, then proceed to select the video they wish to play. It's important to note that the user will also be able to select video files directory.

5.2.2 Working On Input File

Once the video file is selected, the file is processed by three different programs running in parallel (depending on the user machine capability). One program will be responsible for detecting nudity, sexual figures, and pornography. Another program will be responsible for detecting profanity. The last program will detect violence. All of these programs take video as input and return a list of sets, where each set contains the start and end time of explicit frames or audio.

5.2.3 Removing Explicit Content

After all three programs return lists of timestamps, the video player will utilize these lists based on the True/False value of three buttons on the VideoGuard main screen (violence, profanity, nudity). For example, if all three buttons are ON, then the video player will drop a shutter on the screen whenever a violent and nudity timestamp occurs during playback. Additionally, the video player will beep if a profanity timestamp occurs.

5.3 Conclusion

In conclusion, the proposed approach and methodology for the VideoGuard tool aim to provide an effective solution for filtering out explicit scenes from user-selected videos. The process is outlined in two main steps. In the first step, preparing a robust dataset and training the chosen RNN model. Moving on to the second step, the focus shifts to the user's interaction with the VideoGuard tool.

Chapter 6 High-Level and Low-Level Design

6.1 System Overview

Each action is a composition of distinct movements, whether it be a dance sequence or the act of throwing snowballs. These movements involve various body parts and objects of diverse shapes, sizes, and textures. Actions exhibit specific patterns, such as pixel data, spectrograms, and textual data. Recognizing and understanding these patterns play a crucial role in classifying images or videos associated with particular actions or containing specific objects. The system is meticulously crafted to process video data with the primary objective of classifying whether a video contains inappropriate content and subsequently filtering such scenes. The basic design approach adopted for this project:
Firstly, video frames are utilized to extract features, and the relationships between patterns in consecutive frames are analyzed. This analysis is pivotal for classifying frames and eliminating those that contain inappropriate content.
Secondly, recognizing inappropriate scenes can be challenging when video data lacks sufficient pattern details. To address this challenge, audio data is employed. For instance, identifying a person screaming in pain is achievable through audio analysis. Additionally, detecting foul language involves converting speech to text and understanding the meaning of the spoken words through the relationships between them. This comprehensive approach ensures a robust and effective system for content classification.

6.2 Design Considerations

This section describes many issues that need to be addressed or resolved before attempting to devise a complete design solution.

6.2.1 Assumptions and Dependencies

This section focuses on the assumptions and dependencies considered in the making of videoguard tools.

- Videoguard will be using models trained on live-action movies data so it is assumed videos given by users are not animated videos.
- Videos are not blurred and videos are of at least 24 fps and 10 seconds long inorder to provide sufficient data for detecting patterns.
- A complete chunk of frames will be removed i.e., 100 frames, although the explicit frames are less than 100.
- The dataset should be freed from any noise before classification. The better the preprocessing

techniques, the better the classification results will be.

- If running the tool on your own system speed of filtering of the video depends on the system specifications, if specs are not fulfilling the minimum requirement the filtering process can take a long time or the tool may crash.

6.2.2 General Constraints

Here are some general constraints:

6.2.2.1 Algorithm Limitations

The effectiveness of the tool is constrained by the limitations of the underlying algorithms. No algorithm is perfect, and there will always be cases where inappropriate content is either missed (false negatives) or non-inappropriate content is incorrectly flagged (false positives).

6.2.2.2 Cultural Sensitivity

The tool may struggle with cultural nuances, as what is considered inappropriate can vary widely across different cultures and regions. It may be challenging to develop a one-size-fits-all solution that accurately captures cultural context.

6.2.2.3 Dynamic Content

In order to maintain its effectiveness the tool needs to adapt to dynamically changing content trends. Emerging forms of inappropriate content, or rapidly evolving video creation techniques can be a significant constraint.

6.2.2.4 Resource Intensiveness

Real-time or near-real-time processing of videos can be computationally intensive and lacks accuracy. This constraint involves the availability of sufficient computational resources to handle the processing demands, potentially limiting scalability.

6.2.3 Goals and Guidelines

6.2.3.1 Efficient Content Filtering

- Optimize the filtering process to minimize impact on overall video playback performance.

6.2.3.2 User-Friendly Interface

- Design an intuitive and user-friendly desktop application.
- Prioritize ease of use to make the content filtering process accessible to users of varying technical expertise.

6.2.3.3 Scalability

- Develop a scalable system capable of handling videos of different resolutions and lengths.
- Consider future upgrades and enhancements to accommodate evolving video formats and standards.

6.2.3.4 Customization Options

- Provide users with options to customize the filtering criteria based on their preferences.
- Implement settings for different sensitivity levels to cater to a diverse user base.

6.2.3.5 Modular Architecture

- Adopt a modular design to enhance maintainability and facilitate future updates.
- Separate components for input processing, content analysis, and output rendering.

6.2.3.6 Optimized Algorithms

- Implement efficient algorithms for content analysis to ensure real-time or near-real-time processing of videos.
- Continuously optimize and refine the algorithms for improved accuracy and performance.

6.2.3.7 Compliance with Legal Standards

- Adhere strictly to legal standards and regulations regarding content filtering.
- Regularly update the application to comply with any changes in content filtering laws and regulations.

6.2.3.8 Documentation

- Provide comprehensive documentation for users and developers, outlining the functionality and usage of VideoGuard.

- Include clear instructions for configuration and customization options.

6.2.4 Development Methods

The **iterative process model** [27] is a way of developing tool step by step. At first, we work on the essential requirements, and then we keep adding new features to the software until it's complete. It's like building a big project by breaking it into smaller parts. We chose this model for our project because it allows us to finish different stages one at a time during the semester. We complete each phase before moving on to the next one. One good thing about this model is that we can make changes to the final product before it's done, and we can update the documentation to show those changes.

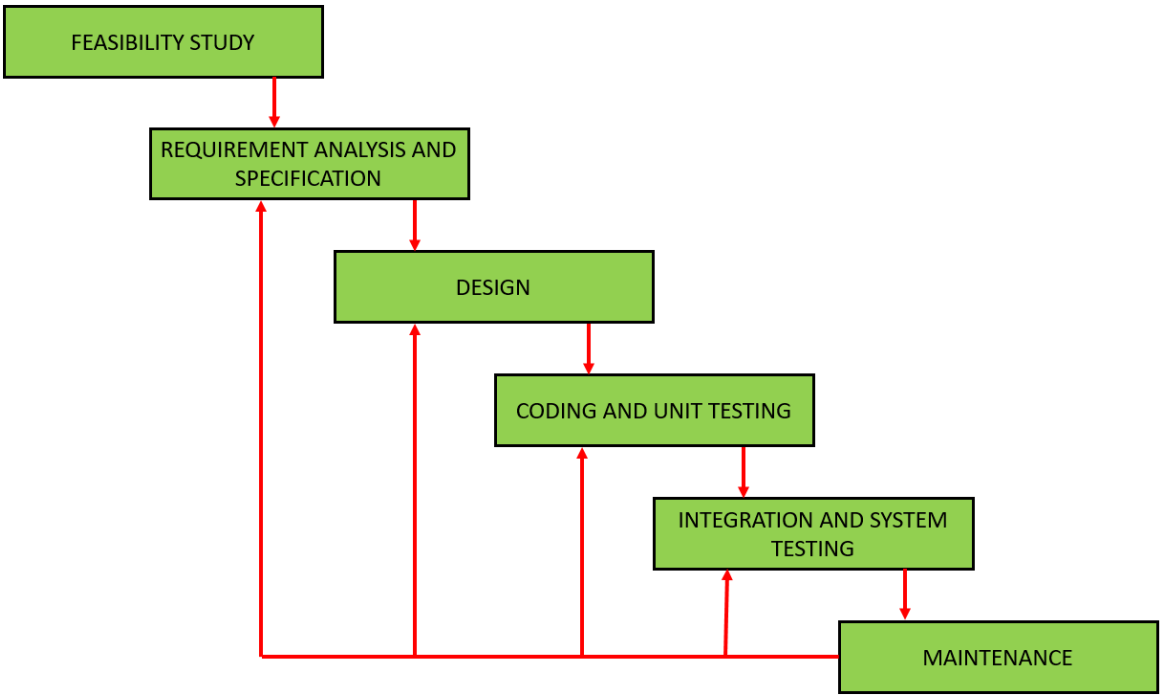

Figure 6.1: Iterative Process Model

This figure shows an iterative model [27] used in developement of VideoGuard.

6.3 System Architecture

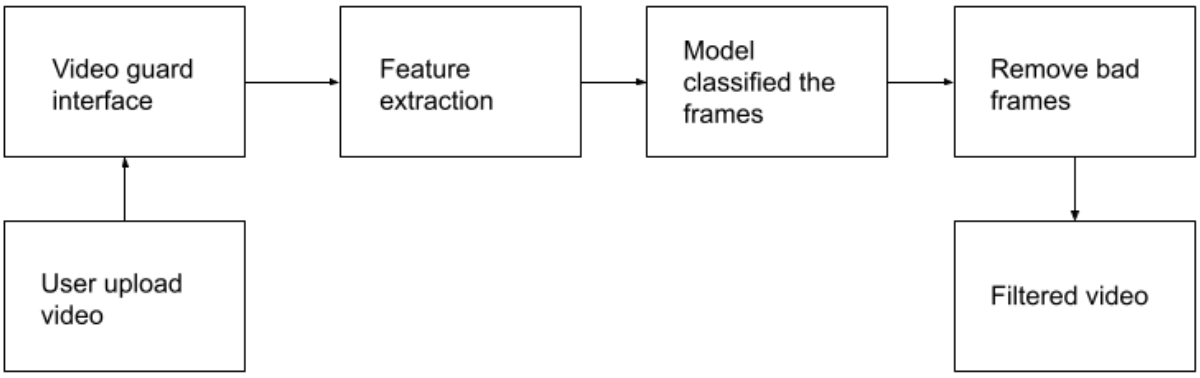

Figure 6.2: A high level structure of VideoGuard

There are three main modules in the VideoGuard Structure, feature extraction, classification and removal.

6.3.1 Feature Extraction

When the user starts playing the video, features will be extracted from video frames in batches. For example, features will be extracted from the first 100 frames (window size), and then the next 100 frames, and so on until the end of the video.

6.3.2 Frames Classification

The extracted features of frames are classified as explicit or non-explicit using an RNN model, also other models, NSFW and Silero. The classification will also be done in batches, i.e., the classification of the first 100 frames (based on extracted features) and then the next 100 frames, and so on.

6.3.3 Removing Frames

Once the frames are classified as explicit or non-explicit, the explicit frames are removed. The system will eliminate the entire window of 100 frames that are classified as explicit. For this purpose, the system will display either blurred frames or skip them entirely, depending on user preferences.

6.4 Architectural Strategies

We will be focusing on following architectural strategies when developing our tool:

6.4.1 Python language

Python for training models to detect inappropriate content and use of Electron JS, HTML, CSS AND JavaScript to develop the tool.

6.4.2 Extensible

We plan to keep the system easily extensible in the future if we want to shift to a client-server model and create a chrome extension.

6.4.3 Error detection and recovery

- **Frame Interpolation:** Use frame interpolation techniques to replace inappropriate frames. This involves making blured frames between existing frames to create a smoother transition.

6.4.4 Memory management policies

- **Memory Pooling:** Use memory pooling techniques to allocate and deallocate memory efficiently. This involves pre-allocating a pool of memory blocks of fixed sizes, reducing fragmentation and improving memory reuse.
- **Parallel Processing and Multithreading:** Leverage parallel processing and multithreading to distribute memory-intensive tasks across multiple threads, optimizing resource utilization.

6.4.5 User Interface paradigms

6.4.5.1 Intuitive Controls

- **Simple Controls:** Keep the controls for video playback, processing, and settings simple and easy to understand. Use familiar icons and labels to enhance user comprehension.
- **Interactive Elements:** Implement interactive elements, such as sliders for video navigation or toggles for enabling/disabling specific features.

6.4.5.2 User Assistance

- **Guided Workflows:** Design guided workflows to assist users in setting up and using the video guard features. Provide tooltips or contextual help to explain complex functionalities.

6.4.5.3 Customization Options

- **Settings Panel:** Include a settings panel that allows users to customize parameters related to video processing, and other application-specific settings.

6.4.5.4 Responsive Design

- **Compatibility:** Ensure that the UI is responsive and compatible across different devices and screen sizes.

6.4.5.5 Accessibility

- **Text and Color Contrast:** Pay attention to text readability and color contrast to ensure accessibility for users with visual impairments.

6.5 Domain Model/Class Diagram

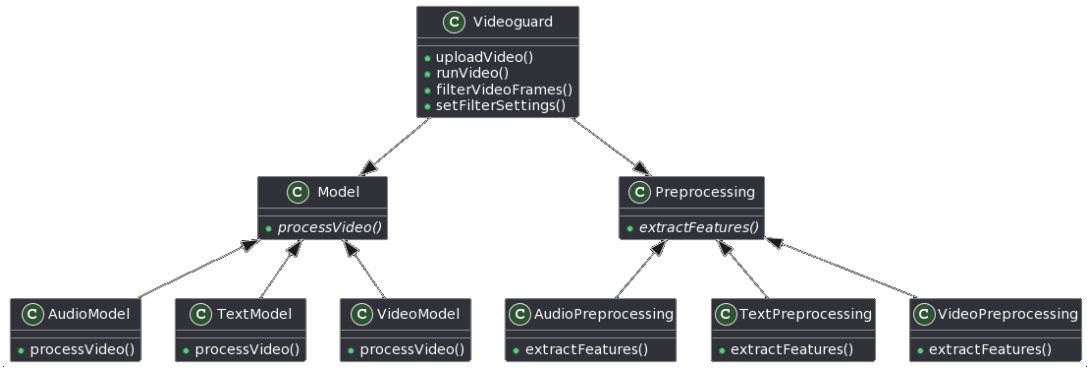

Figure 6.3: Domain/Model Class Diagram

This figure shows the class diagram of VideoGuard.

6.6 Sequence Diagrams

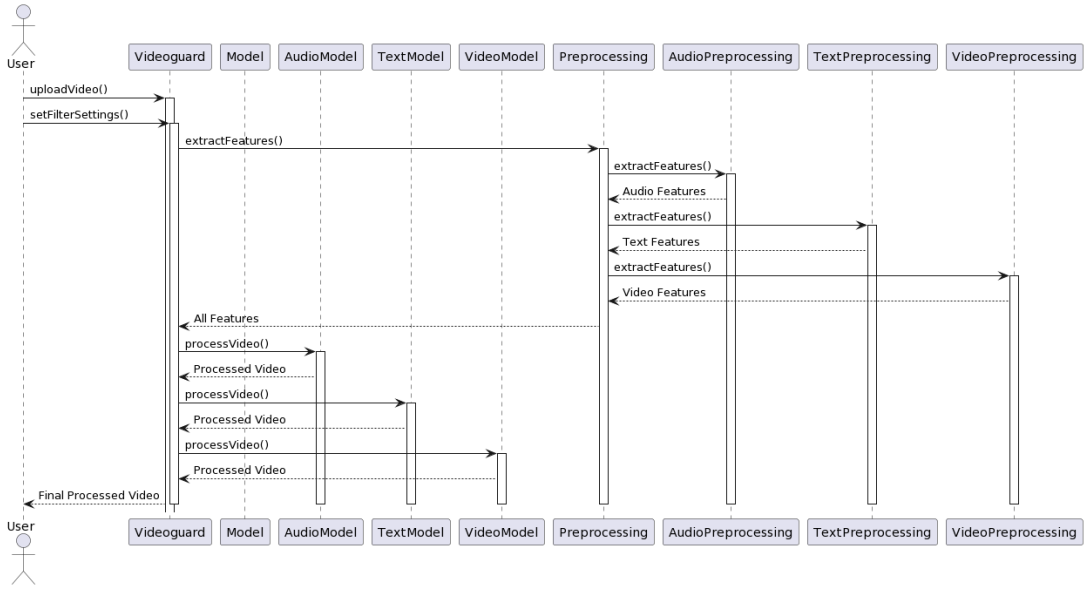

Figure 6.4: Sequence Diagram

This figure shows sequence diagram of VideoGuard.

6.7 Policies and Tactics

This section focuses on the design policies and/or tactics that will be followed throughout the implementation of this project. Following are the policies and/or tactics adopted:

- We are collecting our own data by scraping videos from Youtube, Instagram, Facebook and other similar social media apps or by directly downloading movies.
- Using python to implement pre-processing, feature extraction and classification of the pattern data.
- For training the model at faster speed, we will be using google colab GPUs.
- Electron JS, HTML, CSS and JavaScript is used for making GUI.
- The dataset will be divided into train and test data. The model will be first trained on the train data and than tested on test data. We will use 80% data for training and 20% for testing.

6.8 Conclusion

This chapter explains VideoGuard, a tool like a filter for videos to keep them family-friendly. It checks for inappropriate content by looking at video patterns. The goal is to make it easy for families to watch movies without worry. Parents can control what their kids watch, and users can set filters for what they find inappropriate. VideoGuard is built step by step, using Python and Google Colab for training, and Electron JS for the tool. Testing involves splitting data, and data is collected from YouTube, social and movies media for improvement.

Chapter 7 Implementation and Test Cases

This chapter will exclusively focus on the implementation of the prototype or the work we have done so far, with the inclusion of test cases to follow later.

7.1 Implementation

This section discusses the implementation of the prototype that we have developed so far.

7.1.1 Dataset Collection

We collected around 3000 video clips containing various actions, including gunshots, stabbing, and strangling. Then, we extracted the features from the videos using ResNet50, a large CNN model trained on the ImageNet dataset.

7.1.2 Feature Extraction

After data collection, we used ResNet50 to extract features from each frame of the video clips.

7.1.3 Pre-trained Models

Besides data collection and training our own RNN, we utilized pretrained models to enhance the overall accuracy of our tool. The first pretrained model we used is trained on the NSFW dataset, classifying a frame or image into neutral, porn, or sexual classes. This model is employed in one of our three programs (nudity, violence, and profanity). Based on the classification of individual frames, we calculate the timestamp of that frame by dividing the frame number by fps. Additionally, we utilized Silero for converting speech to text, and then applied NLP to detect foul words in the text. NLP is employed to detect foul words based on the context of the line. We feed a 5-second audio segment to Silero and then check for foul words in the resulting text. If no words are found, we simply advance to the next 5 seconds. If a word is found, we advance 1 second forward and then again search for foul words in a 5-second window. If no words are found, then the foul word was within the 1 second that we advanced, so we mark that 1 second. We also used models to detect weapons in a frame, aiding in the accurate detection of violent scenes.

7.1.4 Models Architecture

For now, we have designed a RNN model—a neural network tailored for processing sequential data, particularly well-suited for tasks involving video or image sequences. The input is configured to handle

batches of data, where each batch contains 100 frames. Each frame comprises 4x4 grids with 2048 channels, resulting in a NumPy array shape of (100, 4, 4, 2048), as mentioned earlier. The model employs a TimeDistributed layer to independently flatten the spatial dimensions of each frame. Following this, three LSTM layer with 64 units captures temporal patterns in the sequence. The activation function used in this layer is the hyperbolic tangent (tanh). The final layer is a Dense layer with a single neuron and a sigmoid activation function, tailored for producing binary classification outputs. The model is trained using the Adam optimizer and optimized for binary crossentropy loss, with accuracy serving as the performance metric.

7.1.5 Training

We trained the model on 80% of our 3000-video dataset, as mentioned earlier. To expedite the training process, we utilized the GPUs available on Google Colab. Currently, the model has been trained for 50 epochs; however, this may be subject to change as we plan to incorporate thousands of videos into our dataset. We trained CNN model with same 80% and 20% split.

7.1.6 Testing

We tested our trained RNN model on the remaining 20% of the data that was set aside during the training process. The obtained accuracy is 81%, but it's important to note that this value may change as we continue to train the model with additional videos on a daily basis.

7.1.7 Testing on Videos

The input video is processed in 100-frame windows. Initially, we pass each window to the feature extractor, and then the extracted features are fed into our RNN model, which determines whether the 100 frames contain violent content. If violence is detected.

7.2 Test Cases

Table 7.1: Video Clipping Function

The table shows the test case of clipping function that make small clips of larger videos.

Clipping Component			
Test Case ID:	*1*	**QA Test Engineer:**	*M.Raza Mehdi*
Test case Version:	*1.0*	**Reviewed By:**	*Saif Ur Rehman*
Revision History:	-		
Objective:	*To verify that the clips created contain 100 frames and are valid.*		
Product/Ver/ Module:	*VideoGuard v1.0*		
Environment:	*Windows 8 or higher and python 3.9.*		
Assumptions:	*The video file path is correct.*		
Pre-Requisite:	*The video contains violence scenes.*		
Step No.	**Execution description**	**Procedure result**	
1	*Run the Clipping Function.*	*Small clips start appearing in output directory.*	
2	*Check if each clip has 100 frames.*	*passed*	
Comments: The test case passed, which mean training clips are valid.			
Passed ☐ **Failed** ☐ **Not Executed**			

Table 7.2: Model Training

The table shows the test case model training.

Model			
Test Case ID:	*2*	**QA Test Engineer:**	*M.Raza Mehdi*
Test case Version:	*1.0*	**Reviewed By:**	*Saif Ur Rehman*
Revision History:	-		
Objective:	*To verify that the trained model working perfectly.*		
Product/Ver/ Module:	*VideoGuard v1.0*		
Environment:	*Windows 8 or higher and python 3.9.*		
Assumptions:	*The model is trained on training dataset.*		
Pre-Requisite:	*The test video is valid.*		
Step No.	**Execution description**	**Procedure result**	
1	*model predicts the labels.*	*passed*	
2	*Check if labels are valid*	*passed*	
Comments: The model is running as expected.			
Passed ☐ **Failed** ☐ **Not Executed**			

Table 7.3: Video Selection

The table shows the test case of video selection.

Video Selection Component			
Test Case ID:	*3*	**QA Test Engineer:**	*Saadan Atif*
Test case Version:	*1.0*	**Reviewed By:**	*Saif Ur Rehman*
Revision History:	-		
Objective:	*To verify that the video selection working fine.*		
Product/Ver/ Module:	*VideoGuard v1.0*		
Environment:	*Windows 8 or higher and python 3.9.*		
Assumptions:	*The selected video is on local directory and has valid formate.*		
Pre-Requisite:	*The VideoGuard app is running on the system*		
Step No.	**Execution description**	**Procedure result**	
1	*Click the open file button.*	*passed*	
2	*Only video files show up.*	*passed*	
3	*Select a video file.*	*passed*	
Comments: The video file selection is working as expected.			
Passed ☐ **Failed** ☐ **Not Executed**			

Table 7.4: Player Functions

The table shows the test case of video player functions.

Video Player Component			
Test Case ID:	*4*	**QA Test Engineer:**	*Saadan Atif*
Test case Version:	*1.0*	**Reviewed By:**	*Saif Ur Rehman*
Revision History:	-		
Objective:	*To verify that the video player functions properly.*		
Product/Ver/ Module:	*VideoGuard v1.0*		
Environment:	*Windows 8 or higher and python 3.9.*		
Assumptions:	*The video is already selected.*		
Pre-Requisite:	*The VideoGuard app is running on the system*		
Step No.	**Execution description**	**Procedure result**	
1	*Click the play button.*	*passed*	
2	*Click the pause button.*	*passed*	
3	*Drag the slidder.*	*passed*	
Comments: All functions of video player are working.			
Passed ☐ **Failed** ☐ **Not Executed**			

Table 7.5: Audio Filter

The table shows the test case of audio filter.

Video Filter Component			
Test Case ID:	*5*	**QA Test Engineer:**	*M.Raza Mehdi*
Test case Version:	*1.0*	**Reviewed By:**	*Saif Ur Rehman*
Revision History:	-		
Objective:	*To verify that the audio filter is working.*		
Product/Ver/ Module:	*VideoGuard v1.0*		
Environment:	*Windows 8 or higher and python 3.9.*		
Assumptions:	*The video is already selected.*		
Pre-Requisite:	*The VideoGuard app is running on the system*		
Step No.	**Execution description**	**Procedure result**	
1	*Apply audio filter.*	*passed*	
2	*Profanity detected.*	*passed*	
3	*Words replaced with beep sound.*	*passed*	
Comments: The audio filter is working as expected.			
■ **Passed** □ **Failed** □ **Not Executed**			

Table 7.6: Nudity Filter

The table shows the test case nudity filter.

Nudity Filter Component			
Test Case ID:	*6*	**QA Test Engineer:**	*M.Raza Mehdi*
Test case Version:	*1.0*	**Reviewed By:**	*Saif Ur Rehman*
Revision History:	-		
Objective:	*To verify that the nudity filter is working.*		
Product/Ver/ Module:	*VideoGuard v1.0*		
Environment:	*Windows 8 or higher and python 3.9.*		
Assumptions:	*The video is already selected.*		
Pre-Requisite:	*The VideoGuard app is running on the system*		
Step No.	**Execution description**	**Procedure result**	
1	*Apply nudity filter.*	*passed*	
2	*nudity detected.*	*Failed*	
3	*nudity scenes are blurred.*	*Failed*	
Comments: The nudity filter is working as expected.			
□ **Passed** ■ **Failed** □ **Not Executed**			

Table 7.7: Violence Filter

The table shows the test case violence filter.

Violence Filter Component			
Test Case ID:	*7*	**QA Test Engineer:**	*Saadan Atif*
Test case Version:	*1.0*	**Reviewed By:**	*Saif Ur Rehman*
Revision History:	-		
Objective:	*To verify that the violence filter is working.*		
Product/Ver/ Module:	*VideoGuard v1.0*		
Environment:	*Windows 8 or higher and python 3.9.*		
Assumptions:	*The video is already selected.*		
Pre-Requisite:	*The VideoGuard app is running on the system*		
Step No.	**Execution description**	**Procedure result**	
1	*Apply violence filter.*	*Passed*	
2	*violence detected.*	*Failed*	
3	*violence scenes are blurred.*	*Failed*	
Comments: The violnce filter is working as expected.			
☐ **Passed** ■ **Failed** ☐ **Not Executed**			

Table 7.8: Filter Options

The table shows the test case violence filter.

Filter Component			
Test Case ID:	*8*	**QA Test Engineer:**	*Saadan Atif*
Test case Version:	*1.0*	**Reviewed By:**	*Saif Ur Rehman*
Revision History:	-		
Objective:	*To verify that the different filter types are working.*		
Product/Ver/ Module:	*VideoGuard v1.0*		
Environment:	*Windows 8 or higher and python 3.9.*		
Assumptions:	*The video is already selected and violence or nudity filter is applied*		
Pre-Requisite:	*The VideoGuard app is running on the system*		
Step No.	**Execution description**	**Procedure result**	
1	*Select blur filter.*	*passed*	
2	*Select black filter.*	*passed*	
Comments: The different filter types are working as expected.			
■ **Passed** ☐ **Failed** ☐ **Not Executed**			

Table 7.9: Directory Selection

The table shows the test case of selecting a directory.

Directory Component			
Test Case ID:	*9*	**QA Test Engineer:**	*Saadan Atif*
Test case Version:	*1.0*	**Reviewed By:**	*Saif Ur Rehman*
Revision History:	-		
Objective:	*To verify that the video directory is loaded to the app.*		
Product/Ver/ Module:	*VideoGuard v1.0*		
Environment:	*Windows 8 or higher and python 3.9.*		
Assumptions:	*The directory contains some video files.*		
Pre-Requisite:	*The VideoGuard app is running on the system*		
Step No.	**Execution description**	**Procedure result**	
1	*Click folder button.*	*passed*	
2	*Select video directory.*	*passed*	
Comments: The video directory is loaded to app as expected.			
Passed ☐ **Failed** ☐ **Not Executed**			

Table 7.10: Manual Filter

The table shows the test case of manual filtering.

Manual Filtering Component			
Test Case ID:	*10*	**QA Test Engineer:**	*Saadan Atif*
Test case Version:	*1.0*	**Reviewed By:**	*Saif Ur Rehman*
Revision History:	-		
Objective:	*To verify that the manual filtering feature is working.*		
Product/Ver/ Module:	*VideoGuard v1.0*		
Environment:	*Windows 8 or higher and python 3.9.*		
Assumptions:	*The video is already selected.*		
Pre-Requisite:	*The VideoGuard app is running on the system*		
Step No.	**Execution description**	**Procedure result**	
1	*Add start time.*	*passed*	
2	*Add end time.*	*passed*	
3	*Select a filter.*	*passed*	
2	*Click apply button.*	*passed*	
Comments: The manual filtering feature works as expected.			
Passed ☐ **Failed** ☐ **Not Executed**			

7.3 Test Metrics

Table 7.11: Test case Matrics

This table shows the total number of test cases we have developed for our project and how many of them passed.

Metric	Value
Number of Test Cases	10
Number of Test Cases Passed	8
Number of Test Cases Failed	2
Test Case Defect Density	(2 * 100) / 8 = 25%
Test Case Effectiveness	(2 * 100) / 4 = 50%

7.4 Conclusion

To conclude, this is just the beginning. From gathering the right data to selecting the best model, every single achievement is crucial for our tool. In this chapter, we briefly discussed what we have developed so far, how we have collected the data, and how the features are extracted. We also mentioned what we are training with these features. Additionally, we discussed the pre-trained open-source models that we are using to enhance the capability of our tool. The upcoming chapters will briefly discuss the results we have achieved and outline our plans for the future.

Chapter 8 User Manual

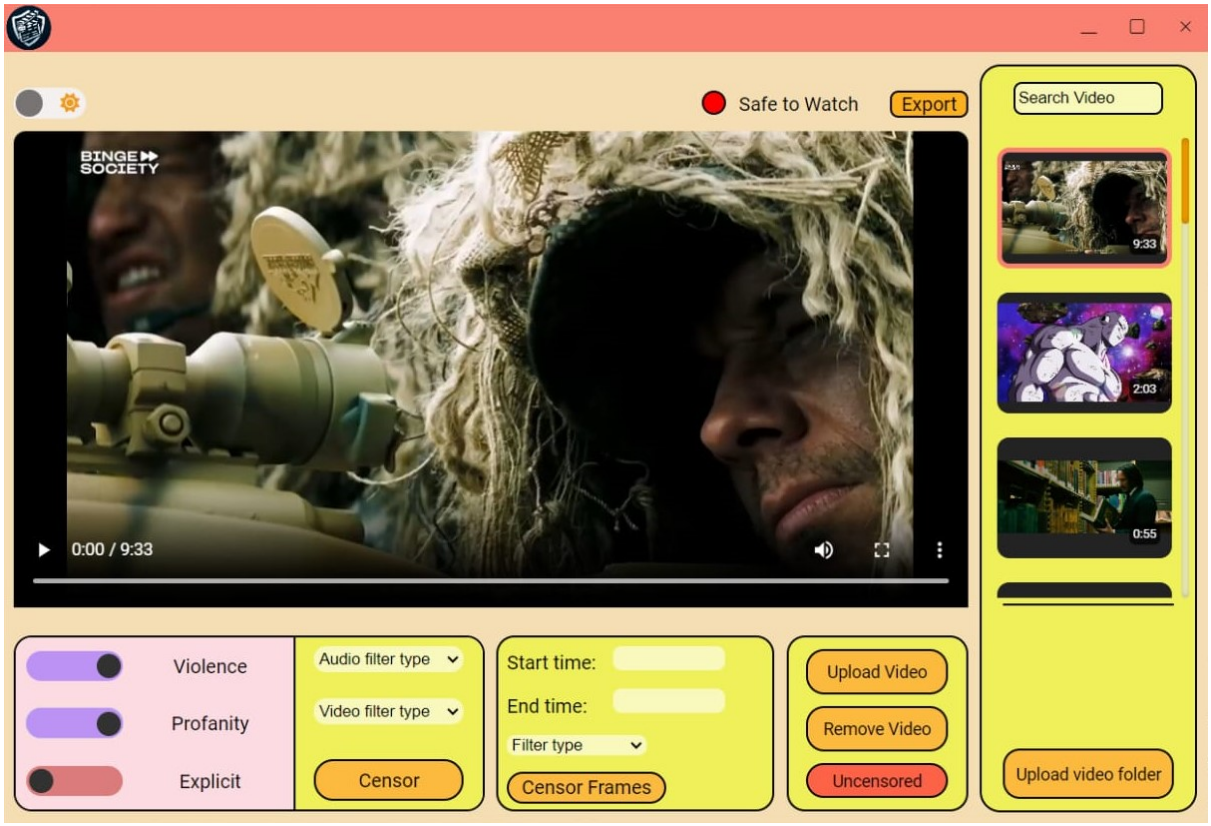

Figure 8.1: A light themed view of VideoGuard

The GUI has an option to upload video and the settings tab contains the filters and level of filtering the user will after the selected video has been processed.

8.0.1 Selecting a video file

Click Upload video button to open file manager. Select the video you want to play and click open.

8.0.2 Processing

Wait until safe to watch indicator turns green from red.

8.0.3 Applying filters

After processing, a filter column will appear with three types of filters, profanity, violence and explicit. Toggle the prefered one.

8.0.4 Playing the video

After selecting the filter, click play button to start playing the video.

8.0.5 Upload folder

Click on upload folder button if you want to load a video directory in the player. A file manager will open, select the folder and click open, all the video files in that folder will be displayed in the side panel.

8.0.6 Remove Video

Click on the remove video button to remove a loaded video from the player.

8.0.7 Change Theme

Toggle theme button on top left corner to switch between light and dark themes.

Chapter 9 Experimental Results and Discussion

This chapter presents experimental results conducted on the prototype we have developed so far.

9.1 Experiment No.1

Firstly, we tested the prototype on unseen data. We collected an additional 180 video clips (in addition to the test data) and passed them through our model, yielding promising results. Out of the 180 clips, our model correctly classified 149, of which 111 were violent and 38 were non-violent. The remaining 31 clips were misclassified, with 20 being non-violent and 11 being violent.

Table 9.1: Confusion Matrix of Experiment No.1

This is the confusion matrix of the experiment conducted on prototype.

	Predicted Non-Violent	**Predicted Violent**
Actual Non-Violent	38	20
Actual Violent	11	111

9.2 Experiment No.2

After testing the model on clips, we applied it to longer videos, and the model successfully detected violent frames within the videos. From this second experiment, we concluded that the window size should be smaller, around 50 frames, rather than 100 frames. This adjustment is necessary because videos with violent frames at frequent intervals were incorrectly classified as completely violent, even though there were some frames without violence. By the end of this experiment, we generated an idea to address this issue without reducing the window size, which we briefly discussed in the future work section.

9.3 Conclusion

In this chapter, we discussed two experiments conducted on our prototype. In the first experiment, we utilized small clips to test the model, and in the second experiment, we applied our model to longer videos. The model performed well on smaller clips; however, for longer videos, it erroneously classified extended intervals as violent, even when they contained some normal scenes.

Chapter 10 Conclusion and Future Work

In today's digital video era, manual content moderation is infeasible given the massive volumes of content being generated. While some companies have developed content filtering APIs, these limit user control. The proposed VideoGuard tool instead aims to empower users with comprehensive control over censoring video content themselves, targeting features like age restrictions, scene removal/blurring, and real-time/saved filtering.

This report outlines the complete process our team followed in creating VideoGuard, an AI-based video censoring tool designed for the Windows platform. We initiated the project with a literature review, studying various articles on similar work conducted in the past. Based on our findings, we determined that transfer learning, specifically using ResNet50 to extract features, would be the optimal methodology for our project. We then trained our RNN model on these extracted features, considering RNN's ability to leverage the temporal nature of video data.

Next, we detailed the features of VideoGuard, which include video selection from the computer directory, setting user preferences for filters, removing or blurring inappropriate frames, muting inappropriate words, and more. Subsequently, we explained the architecture of our project, consisting of three main modules: feature extractor, classifier, and frames remover. Continuing with this report, we delved into the implementation of the prototype we developed—an RNN model trained on 240 video clips, encompassing both violent and non-violent content.

In our future work, we aim to address the window size and explore audio data integration. In the provided prototype, the window size is set to 100 frames, meaning VideoGuard would remove all 100 frames if violence is detected in that window. However, this may not be optimal for the final product, as it could result in the removal of non-violent or non-explicit frames. To overcome this challenge, we plan to implement an overlapping technique. After processing 100 frames, we will move the window forward by only 50 frames. This way, we'll be analyzing 50 new frames while maintaining the temporal context for the RNN. This approach reduces the window size, and if the model labels the initial 100 frames as non-violent, we move forward by 50 frames to identify any newly added violent content.

Although this technique may increase processing time due to reanalyzing some previous frames, it aims to enhance user experience. To implement this, we will test alternative feature extraction techniques to ensure efficiency.

Bibliography

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org. [Accessed: 2023-10-02].

[2] M. Susanty, A. F. Rahman, M. D. Normansyah, A. Irawan, *et al.*, "Offensive language detection using artificial neural network," in *2019 International Conference of Artificial Intelligence and Information Technology (ICAIIT)*, pp. 350–353, IEEE, 2019.

[3] F. Gers and J. Schmidhuber, "Recurrent nets that time and count," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, vol. 3, pp. 189–194, 2000.

[4] E. A. Emon, S. Rahman, J. Banarjee, A. K. Das, and T. Mittra, "A deep learning approach to detect abusive bengali text," in *2019 7th International Conference on Smart Computing & Communications (ICSCC)*, pp. 1–5, IEEE, 2019.

[5] M. J. F. Gales, S. Watanabe, and E. Fosler-Lussier, "Structured discriminative models for speech recognition: An overview," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 70–81, 2012.

[6] W. Han and M. Ansingkar, "Discovery of elsagate: Detection of sparse inappropriate content from kids videos," in *2020 Zooming Innovation in Consumer Technologies Conference (ZINC)*, pp. 46–47, IEEE, 2020.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pp. 630–645, Springer, 2016.

[8] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2016.

[9] C. Demarty, C. Penet, M. Soleymani, and G. Gravier, "Vsd, a public dataset for the detection

of violent scenes in movies: design, annotation, analysis and evaluation," *Multimedia Tools and Applications*, vol. 74, 05 2014.

[10] M. Gruosso, N. Capece, U. Erra, and N. Lopardo, "A deep learning approach for the motion picture content rating," in *2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, pp. 137–142, 2019.

[11] M. Awad, R. Khanna, M. Awad, and R. Khanna, "Support vector machines for classification," *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*, pp. 39–66, 2015.

[12] V. R. Martinez, K. Somandepalli, K. Singla, A. Ramakrishna, Y. T. Uhls, and S. Narayanan, "Violence rating prediction from movie scripts," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, pp. 671–678, 2019.

[13] H. S. Lyn, S. Mansor, N. AlDahoul, and H. A. Karim, "No. 5 convolutional neural network-based transfer learning and classification of visual contents for film censorship," *Journal of Engineering Technology and Applied Physics*, vol. 2, no. 2, pp. 28–35, 2020.

[14] J. Su, P. Her, E. Clemens, E. Yaz, S. Schneider, and H. Medeiros, "Violence detection using 3d convolutional neural networks," in *2022 18th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–8, 2022.

[15] M. Cheng, K. Cai, and M. L. RWF, "Rwf-2000: An open large scale video database for violence detection," in *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 4183–4190, 2021.

[16] T. Aremu, L. Zhiyuan, R. Alameeri, M. Khan, and A. El Saddik, "Ssivd-net: A novel salient super image classification & detection technique for weaponized violence," 2023.

[17] Y. Itcher, *Real-Time Detection of Violent Crowd Behavior*. Open University of Israel, 2013.

[18] P. V. de Freitas, P. R. Mendes, G. N. dos Santos, A. J. G. Busson, Á. L. Guedes, S. Colcher, and R. L. Milidiú, "A multimodal cnn-based tool to censure inappropriate video scenes," *arXiv preprint arXiv:1911.03974*, 2019.

[19] D. Moreira, S. Avila, M. Perez, D. Moraes, V. Testoni, E. Valle, S. Goldenstein, and A. Rocha, "Multimodal data fusion for sensitive scene localization," *Information Fusion*, vol. 45, pp. 307–323, 2019.

[20] H. Chen, W. Xie, A. Vedaldi, and A. Zisserman, "Vggsound: A large-scale audio-visual dataset,"

in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 721–725, IEEE, 2020.

[21] M. Huang, "Theory and implementation of linear regression," in *2020 International Conference on Computer Vision, Image and Deep Learning (CVIDL)*, pp. 210–217, 2020.

[22] K. Jani, M. Chaudhuri, H. Patel, and M. Shah, "Machine learning in films: an approach towards automation in film censoring," *Journal of Data, Information and Management*, vol. 2, pp. 55–64, March 2020.

[23] P. Cerda and G. Varoquaux, "Encoding high-cardinality string categorical variables," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 3, pp. 1164–1176, 2022.

[24] A. Chaudhari, P. Davda, M. Dand, and S. Dholay, "Profanity detection and removal in videos using machine learning," in *2021 6th International Conference on Inventive Computation Technologies (ICICT)*, pp. 572–576, 2021.

[25] Silero AI Team, "Silero speech-to-text models." https://pytorch.org/hub/snakers4_silero-models_stt/. [Accessed 2024-01-10].

[26] N. Aldahoul, H. A. Karim, M. H. L. Abdullah, A. S. B. Wazir, M. F. A. Fauzi, M. J. T. Tan, S. Mansor, and H. S. Lyn, "An evaluation of traditional and cnn-based feature descriptors for cartoon pornography detection," *IEEE Access*, vol. 9, pp. 39910–39925, 2021.

[27] GeeksforGeeks, "Software engineering iterative waterfall model." https://www.geeksforgeeks.org/software-engineering-iterative-waterfall-model/. [Accessed 2023-11-04].