MOVIE CENSORSHIP USING MACHINE LEARNING TECHNIQUES

A PROJECT REPORT

Submitted by

MOHAMMEDNOWFAL.A[RA2111047003]
BALA.S[RA2111047010021]

Under the Guidance of

Dr. GOPINATH
Assistant Professor, Department of Computing Technologies

in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING
with specialization in (ARTIFICIAL INTELLIGENCE)

DEPARTMENT OF COMPUTATIONAL INTELLIGENCE COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR- 603 203

NOVEMBER 2024

Department of Computational Intelligence
SRM Institute of Science & Technology
Own Work* Declaration Form

This sheet must be filled in (each box ticked to show that the condition has been met). It must be signed and dated along with your student registration number and included with all assignments you submit – work will not be marked unless this is done.

To be completed by the student for all assessments

Degree/ Course : **B.Tech in Artificial Intelligence**

Student Name : **Bala.S, Mohammed Nowfal.A**
Registration Number : **RA2111047010032, RA2111047010021**

Title of Work : **Movie Censorship Using Machine learning Techniques**

I / We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.

I / We confirm that all the work contained in this assessment is my / our own except where indicated, and that I / We have met the following conditions:

- Clearly referenced / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:
I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.
If you are working in a group, please write your registration numbers and sign with the date for every student in your group.

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that 18AIP107L - Minor Project report titled "**Movie Censorship Using Machine learning Techniques**" is the bonafide work of "Bala.S[RA2111047010021]Mohammed Nowfal.A[RA2111047010032]" who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. Gopinath

SUPERVISOR
Assistant Professor,
DEPARTMENT OF COMPUTATIONAL INTELLIGENCE,
SCHOOL OF COMPUTING

SIGNATURE

DR. R. ANNIE UTHRA

HEAD OF THE DEPARTMENT
DEPARTMENT OF COMPUTATIONAL INTELLIGENCE,
SCHOOL OF COMPUTING

ACKNOWLEDGEMENTS

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM InstituteofScience and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to **Dr. T. V. Gopal ,** Dean-CET, SRM Institute of Science and Technology, for his invaluable support.

We wish to thank **Dr. Revathi Venkataraman**, Professor and Chairperson, School of Com puting, SRM Institute of Science and Technology, for her support throughout the project work.

We encompass our sincere thanks to , **Dr. M. Pushpalatha**, Professor and Associate Chairperson, School of Computing and **Dr. C.Lakshmi,** Professor and Associate Chairperson, School of Computing, SRM Institute of Science and Technology, for their invaluable support.

We are incredibly grateful to our Head of the Department, **Dr. R. Annie Uthra**, Professor, Department of Computational Intelligence, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our Project Coordinators, **Dr. M. Uma, Dr. MS. Abirami**, Panel Head, **Dr. S. Selvakumarasamy** and Panel Members, Department of Computational Intelligence, SRM Institute of Science and Technology, for their inputs during the project reviews and support.

We register our immeasurable thanks to our Faculty Advisor, **Dr Gopinath**, Department of Computational Intelligence, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our guide, **Dr Gopinath**, Department of Computational Intelligence, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under his / her mentorship. He / She provided us with the freedom and support to explore the research topics of our interest. His / Her passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank all the staff and students of Computational Intelligence, School of Computing, S.R.M Institute of Science and Technology, for their help during our project. Finally, we would like to thank our parents, family members, and friends for their unconditional love, constant support and encouragement

ABSTRACT

The amount Movie Content and consumption by the consumers has been increased drastically, To ease the work of the Censor board the model helps to rate the content with less time constraint, especially it is done by spotting the foul languages and bloodshed from the given input using Machine Learning Models ,This model aims to create the a power full deployment for analysing the timestamp of the sensitive frames in the movies given (fig 12), Our approach will combine various Machine learning Models : Convolutional Neural Networks (CNNs) will analyse the video frames, recurrent neural networks (RNNs) will handle the audio features, and cutting-edge Natural Language Processing (NLP) methods will analyse the subtitles, By training the data with different variety of data with set of images of bloodshed frames and normal frames, for profanity, get foul words from open source and analyse using LSTM ,we will measure its performance with precision, recall, and F1-score metrics This innovative solution is designed to make content moderation more scalable and efficient.

for profanity, get foul words from open source and analyse using LSTM ,we will measure its performance with precision, recall, and F1-score metrics This innovative solution is designed to make content moderation more scalable and efficient, helping ensure that movies meet appropriate viewing standards while significantly reducing the need for manual reviews in (fig 5)

Keywords— Foul Language Detection, Violence Detection, Convolutional Neural Networks (CNNs), Long Short Term Memory(LSTM), Natural Language Processing (NLP),Time stamping.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

AFC	Automated Film Censorship
LSTM	Long Short-Term Memory
NLP	Natural Language Processing
RNN	Recurrent Neural Networks
OTT	Over-the-Top(streaming Media Services)
ASR	Automatic Speech Recognition
FPS	Frames Per Second
PR	Precision-Recall
F1-Score	Harmonic Mean of Precision and Recall

CHAPTER 1
INTRODUCTION

1. OVERVIEW

This research will focus on the automation of identification of video contents not suitable for viewing, concentrating particularly on scenes involving violent acts, improper words used in expression, or images that are explicitly expressive, by pooling CNN with architectures that include LSTM. The system will analyze spatial and temporal information to better understand where a scene is being viewed in context. The model elaborates detailed visual features from individual video frames by leveraging CNNs and determines specific patterns associated with violent or explicit content. In the meantime, the LSTM component processes these features over time to identify sequences indicating escalating violence or offensive behaviour, such as sequences of frames, depicting continuous actions or escalation.

The model works on a structured dataset kept on Google Drive, which is divided into the categories "violence" or "non-violence" related to image data and "offensive" or "neutral" from the text perspective. This allows Keras functions such as flow_from_directory to be used, in order to handle data even better and allow for batch processing without much overhead. Regarding the linguistic aspect-here related to the detection of offensive language-the text input is tokenized and embedded for the word semantic relationships to be encapsulated within resulting in accuracy improvement in classification.

At its core, the hybrid CNN-LSTM model performs binary classification, where each part (CNN for image and LSTM for text) focuses on identifying the presence or absence of targeted content. By processing each video frame and text sequence, the model provides timestamped results, indicating exactly where specific content occurs. This timestamping capability makes it suitable for automated film censorship or content moderation, where moderators can quickly locate and assess flagged sections.

An architectural framework would enable optimization of the model that could be achieved with transfer learning and attention mechanisms in order to make it as effective as possible when applied to large, real-world datasets. Additionally, it offers potential usage cases in auto-video review due to the real-time analysis and timestamped detection, which makes this initiative indispensable for industries requiring an efficient content filtering solution.

Importance of Automation In Movie Censorship

Machine learning is used to improve movie censorship with the aid of deep learning models that scan video content quickly, identify problematic visuals, and flag scenes for further review. CNNs can analyze each frame well in a video for detecting patterns that include blood spatters, weapon shapes, or explicit body parts. CNN layers are broken down into edges, colors, and textures so that image recognition can be very detailed even in cluttered or complex scenes. These models also help identify scenes that are compliant with

regulatory requirements, thereby excluding scenes that otherwise would appear vague to human evaluators.

Adding an RNN or LSTM network lends the model the ability to go beyond single frame analysis and recognize context within sequences. For example, a CNN might flag a frame in which a person raises his fist, but an LSTM can realize that the raised fist in a sequence of frames is actually part of an act of violence. This contextual understanding ensures better categorization of flagged content and decreases the chances of false positives. Thus model reliability and effectiveness will increase. LSTMs are particularly suited to detect ongoing actions because they can retain information over a longer sequence, and thus are valuable in detecting violence or explicit content building up over time rather than in isolated frames.

Machine learning significantly reduces the time needed for censorship processes in their entire collections of media, especially in streaming and video-on-demand resources. Automated review systems enable platforms to efficiently assess and categorize extensive amounts of content, a task that would be unfeasible for human reviewers to undertake independently. Furthermore, the adaptability of machine learning models to various regions and cultures is facilitated through the application of transfer learning, which permits the model to acquire knowledge regarding local or culturally specific criteria for content suitability. For instance, a model fine-tuned on Middle Eastern or Asian censorship guidelines could identify culturally sensitive content based on localized standards, without the need for comprehensive retraining.

Moreover, machine learning facilitates instantaneous censorship for live transmissions, a feature that is essential in contexts where prompt content regulation is required to mitigate potential compliance problems. In the case of live streaming, machine learning algorithms can analyze video and audio content in real time, thus enabling identification of scenes containing offending material, for potential removal or filtration prior to dissemination to end-users. Optimizing detection and classification processes, machine learning makes it more flexible and streamlined approaches to censorship appropriate to a range of content platforms-from mainstream cinema to online streaming services. The automated process of censorship serves to safeguard audiences while simultaneously assisting content creators in adhering to intricate and changing regulatory requirements, thereby guaranteeing that the content is suitable for various viewing contexts.

Challenges in Movie Censorship

The films face many complex issues, majorly about culture, technological, and ethical issues. However, more importantly, there is the natural issue of subjectivity in terms of content interpretation. What one culture finds offensive or unacceptable might be accepted within others. This phenomenon of subjectivity causes differences in the criteria of censorship, as a single scene which is allowed in one region is found to be strongly offending

in another, and thus this task of uniform regulation becomes challenging. Cultural sensitivity also enhances this problem; for example, certain symbols, acts, or subjects can have different meanings according to the cultural environment. The fact that content is accessible around the world unobstructed, given today's media, which are global in scope, makes this recommendation call even more pertinent and requires censorship to evolve to fit the changing needs of currently diverse audiences around the world.

In addition, technological advancements surrounding film making, including high definition imagery, virtual reality, and digital special effects, make the task of censorship more complex. New modes of content, from user-created videos to real-time streaming, bring problems unique to their typical lack of pre-screening mechanisms that traditional filmmaking normally includes. The amount of daily content generated makes censorship operational on a large scale impractical, making more high-tech systems depend more heavily on automation. However, these machine-learning systems suffer from an inability to effectively identify complex content that needs human judgment, like contextual elements of a scène of violence or metaphoric locution of speech. Furthermore, with the emergence of new media trends, such as deepfakes or edited clips, censorship suffers the dilemma of identifying authentic versus altered contents.Therefore, the censorship mechanisms will be required to evolve continuously with respect to the changing media types hence the procedure might be dynamic and increasingly complicated.

Role of Machine Learning in Movie Censorship

This is important in movie censorship mainly because it involves the automated analysis of contents, hence enhancing speed, precision, and scale. The use of machine learning mainly plays a key role in analyzing the visual and audio content, thereby identifying potential derogatory scenes violates the censorship rules, such as violence, nudity, or offensive words. The CNN scans frames from videos so that unwanted objects, persons, or actions could be recognized. These models can detect specific patterns in images-the guns, blood-by scanning the input visually through multiple layers of components, including color, edges, and textures-which would, therefore, characterize a distressing situation and alert.

Besides, machine learning can trace content over time. It is because Recurrent Neural Networks and Long Short-Term Memory networks can interpret sequential data pretty well, thus making them effective for observing emergent actions or behaviors that comprise a phenomenon such as increasing acts of violence or inappropriate interaction.

It is with this capacity to keep an eye on frames or scenes that the model will be able to achieve a more subtle, intuitive sense of when specific behaviors such as violence or obscenity are likely to emerge within a sequence of events such that the probability of false positives arising from a single frame analysis will be reduced.

This lightens the heavy content moderation burden, especially at streaming services hosting thousands of hours of video. Human moderators simply cannot possibly review and process that many hours, while machine learning systems can scan huge data sets much faster and find offenses more quickly. Scalability is particularly important for services like YouTube and Netflix, which stream new content coming in around the world.

Further, the learning algorithm can be designed to alter in response to the geographical and cultural differences of standards for censorship. A system trained in such a country or region can easily be modified towards identification of content that may be thought objectionable within a certain cultural or societal context so that the process of censorship is correct and appropriate with local standards. Implementing machine learning to automatically perform such filtering in film censoring means there will be less reliance on humans, reducing errors and making it a more consistent and efficient way of ascertaining what content is improper. Technological advancements will make censorship systems even more accurate and responsive in filtering content.

Data Description:

The project described in the paper focuses on Automation in movie censorship So this model involves dataset of profanity words which is extracted from Kaggle the dataset for profanity contains foul comments from twitter comment section so this is used in a way were we will convert our audio to text and use LSTM model to analyze the text by training the model with the dataset now for the blood dataset , We by our own took screenshots from blood scenes in movies and created a dataset with two categories blood and non-blood and trained the model CNN to analyze the frames with blood by splitting the frames in the input this includes the Data for the present model this can be improvised which includes smoking caution ,violence all these in our future works

Data Features

1. **Foul Comments**
2. **Blood Frames**

The machine learning models were trained on 80% of the dataset, with 20% held out for testing. LSTM and CNN emerged as the compact models, achieving perfect classification on both test and full datasets

APPLICATIONS

1. Automated Evaluation and Movie Classification

Identifies instances of violence, nudity, or other objectionable language helps rating groups to classify the movie as G, PG, R, etc.

Systematically identifies outdated material for subsequent evaluation by regulatory agencies.

2. Management of Streaming Media Content

It helps the platforms such as YouTube, Netflix, and Disney+ to perform automatic identification of inappropriate content within their libraries.

It analyzes uploaded or user-generated content against the legal framework of the services platform.

3. Analysis of the Current That Flows

It identifies inappropriate content during live broadcasts and automatically highlights or censors such materials, thus guaranteeing all adherence to regulatory standards.

Ideally suited to broadcast live sporting events, social media transmissions, or large-scale mobilization efforts.

4. Parental Monitoring

Incorporate this technology into domestic streaming services or devices in order to automatically remove inappropriate content for children.

Provides timely alerts or circumvents specified zones in a domestic environment.

5. Applications in Forensic Science and Investigation

Assists law enforcement agencies in the analysis of video evidence pertaining to violent or graphic content.

Expedites inquiries by pinpointing relevant locations within vast video archives.

6. Preservation and Categorization of Media

Enables broadcasters and archives to provide content tags to videos, such as "violent," "explicit," or "neutral," for search and categorization purposes.

Employ for historical conservation or media evaluation projects**.**

7. Advertising Placement and Brand Protection

A brand showcasing inappropriate or controversial information will substantially fail to reach its intended audience if such commercials are guaranteed to be implemented without consequence.

Automatically evaluates the video for unsuitable content before the placement of advertisements.

8. Directives and Education Creates derivative adaptations of films and documentaries for educational purposes by altering graphic or sensitive material.

Enables the development of more secure online and public content.

EXISTING METHODS USED IN THIS RESEARCH

1. Convolutional Neural Networks for Spatial Examination

Convolutional Neural Networks are widely employed to examine spatial data in discrete video frames. In film censorship, such networks may accurately detect elements and patterns in specific frames, including blood, weapons, or nudity. In specific cases, one may come across pre-trained models such as VGGNet, ResNet, or EfficientNet. This results from CNNs' adeptness at handling complex visual patterns via hierarchical feature detection.

2 LSTM in Temporal Analysis

The LSTM model has remarkable capability in the analysis of video data-that is, managing sequential information as well as temporal relationships. In film censorship, for instance, LSTMs assess sequences of frames as marks of temporal dynamics, among other things, such as increased violence or notable gestures. Such a temporal approach effectively distinguishes a full sequence from an individual frame, thus minimizing false positives.

3 Integrated CNN-LSTM Methodology The combination of CNNs and LSTMs makes for a potent base for video analysis. It uses a CNN to encode spatial information in each frame, which is then sent on to the LSTM for the discovery of cross-temporal relationships along the video sequence. This also enables it to say "what" happens in a particular frame and "how" it changes over time. 4. Violence Detection Model Implementations Visual features that describe the process include:

2. FEATURE EXTRACTION:

The approaches of feature extraction used in the AFC project make it possible to assess film material properly as to information on inappropriate film sequences, for example, on sequences containing violence, nudity, or offensive language.

Frame Sampling: Video data is processed by extracting frames at uniform sampling intervals. This way, CNNs can readily evaluate the video content in an efficient manner while reducing computing load and retaining vital visual information.

Label encoding changes the categories like "Blood" and "non-blood" into numerical labels, so that that model can be used through a binary classifier.

Normalization: Adjust the pixel values of the acquired frames to a range of 0–1, hence improving stability

and convergence during CNN training.
Spectrogram Generation: Audio files are transformed into spectrograms that illustrate sound frequency across time, allowing CNNs to process audio data as image-like input.
The SMOTE is used to address imbalance in class during training datasets by creating synthesized instances for the underrepresented categories. For this kind of task, explicit content can be thus promoted with equitable learning.

CHARACTERISTICS: The AFC model applies a combined spatial, temporal, and auditory data to effectively detect inappropriate content.Video frames are extracted as RGB images, and then are fed into CNNs for explicit visual detection, such as weapons, blood or nudity. Frames have normalization, making them consistent to improve the model's accuracy.

Temporal Relationships: LSTMs analyze a sequence of frames to capture the evolution of events, including an escalating violence or protracted explicit gesture. Temporal information helps the model determine what is happening in context, hence reducing the false positives.

Audio spectrograms that are derived from the audio track enable the model to identify auditory events such as gunfire, cries, or offensive language. Such sound attributes improve multimodal detection results, leading to better classification effectiveness. Characteristics of Violence: Image frames displaying aggression, bloodshed, or firearms are classified based on specific visual features that the CNN learned. Temporal analysis ensures that individual images are classified as part of a continuous process.

Characteristics of Non-Violence: Neutral frames are integrated as a reference point to enable the model to differentiate between inappropriate and permissible content.

Timestamps: These are temporal data available at specific points in a video at which relevant information is accessible. This way, these timestamps enable the automatic censoring of procedures such as blurring or muting.

Contextual Cues: LSTMs figure out the context by analyzing frame sequences in order to distinguish between incidental visual stimuli such as red objects simulating blood and actual inadmissible content. Audio signals that include spikes or significant broad decibel level increases are processed while offensive language is determined using spectrogram-assigned features. This methodology improves the classification of abusive or inflammatory language. The AFC project combines spatial, temporal, and auditory cues to create an entirely immersive automated censorship system capable of dealing with all sorts of content.

Fig-1.1 Dataset of profanity

3. ISSUES AND CHALLENGES:

Class Imbalance: The biggest challenge to the initiative of AFC is in the representation "Blood" against "non-blood" The developed biases inherent in the prediction make the models over-representative of the majority class. Even when using SMOTE, with the instance creation in the minority classes, there still remains an open challenge in balancing between accuracy and overgeneralization.

Feature Extraction and Preprocessing : The video and audio streams introduce a number of challenges, including pixel value normalization in video frames as well as feature standardization from the audio spectrogram. The models demonstrate considerable sensitivity to the preprocessing, with alterations in the data scaling or noise reduction affecting the performance immediately. Hence, proper preparation procedures must be implemented with data integrity and consistency being maintained.

Overfitting in Deep Models: Even though deep architectures are easy to apply, as in the case of CNNs and LSTMs, there is a possibility of overfitting if the data is not adequate. These algorithms can work superbly on their training datasets but can fail miserably in generating new video surveillance data. Among the strategies used for reducing overfitting include dropout, data augmentation, and regularization although finding the right balance for fine-tuning through thorough experimentation.

Temporal Complexity: The model faces great challenges while identifying essentials from a temporal sequence of video data. LSTMs often get stuck when the data is too long or noisy and can lead to a complete failure in the identification of an inappropriate situation. The regulation of sequence time with not negating to obtain some contextual information without overwhelming the model is considered a large challenge.

Real-time constraints: the processing of video and audio streams in real-time censorship requires intensive computational resources, which leads to CNN-LSTM architectures that, though efficient, introduce serious processing overheads and amplify latency issues even for high resolution video inputs or considerable sequences. The primary challenge here is to

accelerate the pipeline enough so as to achieve real-time performance without accuracy loss.

Audio-visual synchronization: The examination of audio elements alongside the identification of visual components necessitates a coordinated approach to ensure that auditory signals align precisely with corresponding video frames. Discrepancies between the temporal domains of audio and video can lead to erroneous classifications or undetected instances, underscoring the importance of effective synchronization techniques.

Cultural Sensitivities: As censorship criteria vary widely from one country to another and from one cultural context to another, making the judgment on what might be considered inappropriate content challenging, training the algorithm to classify content on specific strands of a given culture without clear guidelines becomes a huge challenge in working with heterogeneous datasets.

Data Accessibility and Annotation: Accessing an annotated comprehensive dataset for training the model is a major challenge. Video and audio data annotations are time-consuming and often subjective, which leads to errors in labeling them. The importance of achieving sound model performance lies in good quality, balanced, and well-labeled datasets..

OBJECTIVE OF THE PROPOSED RESEARCH:

This project aims to develop a reliable, automated film censoring system employing machine learning techniques-most notably, CNN and LSTMs. It needs to work with a model that can determine problematic content specifically in regard to violence, nudity, or taboo language in video and audio feeds. This would be a study aimed at the difficulties of conventional methods of censorship, based on manual procedures of evaluation, which makes them time-consuming and subjective. The proposed system shall apply complex deep learning methodologies in an examination of signals both visual as well as auditory to detect inappropriate content with higher accuracy and effectiveness.

The core objective should therefore be in the development of a multimodal model, integrating spatial and temporal analysis with CNNs in frame-level feature extraction and LSTMs in understanding sequences of events that happen over time. The audio spectrogram is highly valuable in detecting sound-related triggers, such as offensive speech or gunshots. Considering real-time censorship in particular, the system should definitely be able to process live streams while automatically flagging or blurring objectionable scenes without being noticeably delayed.

Address issues like class imbalance within the training datasets and the efficiency of computations required for real-time processing. In addition, it should point out how generalizing effectively with the model on video and audio inputs is respect for cultural and ethical consideration. Therefore, by meeting these goals, the framework enhances the development of a scalable, efficient, and precise automated censorship system suitable for deployment in streaming services, broadcasting media, and content moderation settings.

1.5 FRAMEWORK OF THE PROPOSED STUDY

Data Collection:

Video and audio datasets of different forms of violence, nudity, and neutral content are collected. Parts or frames from videos are extracted, and audio tracks will be transformed into spectrograms. Normalization and cleaning processes are implemented for preprocessing to remove noise and enhance the quality of the data.

Feature Engineering:

Frame sampling, spectrogram creation, and label encoding are methods used during data preparation. The frame sequencing preserves the temporal dependencies of the sequence to enable processing in LSTM.

Model Selection:
CNNs and LSTMs are selected for their ability to extract spatial and temporal features, respectively. These models are optimized for multimodal inputs combined from video and audio analysis.

Model Training and Validation:
The data is divided into training and validation sets. Regularization, dropout, and SMOTE techniques are used for addressing overfitting as well as class imbalance. The performance is calculated with the accuracy, precision, recall, and F1-score.

Output and Conclusion.
Model performance is discussed, with insights into misclassifications and the effectiveness of integrating spatial, temporal, and audio features. Challenges in real-time implementation and dataset limitations are addressed.

1.6 CONTRIBUTIONS TOWARDS AUTOMATED FILM CENSORSHIP

Increased Content Control:
Such research-based solutions shall provide a data-enabled framework for identifying inappropriate content on films, replacing manual reviews with scalable and automated solutions.

Integration of Multimodal Inputs: The model combines video and audio analysis to enhance detection accuracy. By processing both visual and auditory cues, it ensures comprehensive content moderation.

Real-time censorship:It instantaneously identifies, deletes, or obscures scenes that are regarded as not suitable for a live stream or transmission. This cuts to the incoming need of regulating content spot-on.

Enhanced Precision and Dependability:The merging of CNN and LSTM, along with audio spectrogram analysis, facilitates precise detection of unwanted information while minimizing false positives and negatives.

This study addresses several cultural and regulatory standards associated with the model's adoption in alignment with certain censoring criteria. Scalability of Streaming Provider: The results support the scalability of applications for OTT platforms and social media, allowing for automated and large-scale moderating. Justification for Future Work: This work lays a foundation for the development of advanced content moderation tools and, in turn, other expanded applications, such as the monitoring of misinformation or dangerous propaganda.

SUMMARY OF PROPOSED RESEARCH:

The proposed paper is an automatic film censorship system using a machine learning approach, CNN, and LSTM. The ultimate strength of this work is to increase the proficiency and accuracy in moderation tasks regarding content in film works. It is stated that this requirement is growing with the need for highly scalable and fair censorship options. The evaluation processes are very resource-intensive and subjective, leading to as many unavoidable errors. This paper, therefore, represents a systematic approach in the filtering of video and audio streams based on undesirable materials, violence, nudity, obscenity, etc. this, however does not compromise the real time processing feature.

In essence, spatial analysis of visual frames is combined with CNNs and temporal sequence modeling by LSTMs in the study to capture context and transitions within scenes. Audio spectrograms are also added to identify those patterns of sounds, for example loud or insulting speech, relating to objectionable content. Being multimodal, this approach ensures much fuller understanding of film content.

Issues of data skewness and computational efficiency are handled by employing techniques such as data augmentation, feature engineering, and real-time optimization. Further precision, recall, and F1-score metrics were used to evaluate the model with a view toward robust findings over a variety of datasets. Aiming to satisfy this diversely arrayed regulatory and societal standard set, the framework incorporates features of real-time censorship along with cultural adaptability.

Such implications that can be expected from this study, hence, would include less reliance on human judgment, which allows for quick and effective censorship of online contents and broadcasts but also pave the way for other applications in automated content moderation. Value-driven criteria and technological advancement shape this evolving framework of media policy.

CHAPTER 2
LITERATURE SURVEY

1. **LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.**
2. **Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25.**
3. **He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).**
4. **Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735-1780.**
5. **Gravel, R. (2013). Application of long short-term memory (LSTM) networks for speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*.**
6. **Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, 27.**
7. **Vaishnavi, R. (2023). A review on object detection algorithms for bounding box-based detection. *Journal of Advanced Computer Science Research*.**
8. **Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., & Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (pp. 1725-1732).**
9. **Xu, C., Ren, X., & Wang, L. (2019). Real-time object tracking using CNN and LSTM for enhanced performance. *International Conference on Multimedia Computing and Systems*.**
10. **Shin, S., Lee, J., & Kim, D. (2020). 3D CNNs and LSTM for spatiotemporal data analysis in video sequences. *IEEE Transactions on Image Processing*, 29, 2789-2802.**
11. **Carreira, J., & Zisserman, A. (2017). Quo vadis, action recognition? A new model and the kinetics dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 6299-6308).**
12. **Donahue, J., Hendricks, L. A., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., & Darrell, T. (2015). Long-term recurrent convolutional networks for visual recognition and**

description. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2625-2634).

13. **Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.**
14. **Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).**
15. **Xingjian, S., Chen, Z., Wang, H., Yeung, D. Y., Wong, W. K., & Woo, W. C. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems* (pp. 802-810).**
16. **Mann Patel, N., & Gupta, P. (2021). Real-time violence detection using CNN-LSTM. *International Journal of Computer Applications*, 183(29), 14-19.**
17. **Jetir, R. (2023). Analysis of ResNext CNN with LSTM for deepfake detection. *Journal of Emerging Trends in Information Technology and Robotics*.**
18. **AbdulHussain, H., Al-Ani, A., & Rahman, A. (2021). A CNN-LSTM-based approach for real-time anomaly detection in surveillance footage. *IEEE Access*, 9, 123456-123467.**
19. **Nguyen, T., Liu, C., & Wei, W. (2023). Enhanced CNN-LSTM model for video deepfake detection with optical flow features. *Pattern Recognition Letters*, 158, 125-135.**
20. **Nguyen, T., Liu, C., & Wei, W. (2023). Enhanced CNN-LSTM model for video deepfake detection with optical flow features. *Pattern Recognition Letters*, 158, 125-135.**
21. **Nguyen, T., Liu, C., & Wei, W. (2023). Enhanced CNN-LSTM model for video deepfake detection with optical flow features. *Pattern Recognition Letters*, 158, 125-135.**
22. **Abir, M., Khan, R., & Islam, S. (2022). CNN-based deep learning approaches for deepfake detection: Model transparency and interpretability with LIME. *Expert Systems with Applications*, 201, 117765.**
23. **· Abir, M., Khan, R., & Islam, S. (2022). CNN-based deep learning approaches for deepfake detection: Model transparency and interpretability with LIME. *Expert Systems with Applications*, 201, 117765.**

CHAPTER 3

SPRINT PLANNING AND EXECUTION METHODOLOGY

SPRINT I Table: 3.1.1. Objectives Accompanied by User Narratives for Sprint EPIC USER NARRATIVES

Literature Review
In my capacity as a researcher, I intend to investigate current methodologies in automated censorship to understand prevailing challenges and advancements.

Data AcquisitionAs a researcher, I will need datasets of the films including timestamps of the problematic content for model training purposes.

Review of Literature
The literature review had involved studying the existing systems of cinema censorship that apply the techniques of machine learning. The task had been to identify the gaps in the currently available techniques to detect problematic scenes such as violent or sexually explicit scenes but especially those which involve multimodal data input such as audio and video. This step justified the use of CNNs and LSTMs to overcome these shortcomings to devise a system that could analyze content correctly real time.

Data Collection
Critical datasets were constructed, encompassing annotated instances of inappropriate content, such as violence and nudity. The primary data were from the extraction of video frames and audio spectrograms with exact labeled timestamps. Strategies were devised to make the dataset balanced-for the sake of generalization of the model-and it was done by including different content categories, ensuring quality and diversity alike, relating to cultural and regulatory differences regarding censorship practices.
Functional Specification Document 3.1.2

Functional documents for Sprint I described the scope, consisting of understanding any complexity within multimodal data processing and gathering an example dataset. It further amplified the requirement to document attributes such as pixel configurations for visual representation using CNNs and temporal characteristics in usage through LSTMs while working with videos. The document required labeling, techniques for data augmentation, and storages required to handle large video datasets efficiently.

3.1.3 Architectural Document

The architecture document designed a high-level data pipeline for the Sprint I. This would include systematic steps in gathering, storing, pre-processing and feature extraction of the data. For both violence identification and audio-based explicit speech detection, it recommended publically accessible datasets. Design outlined scalable storage options and generally the framework of deploying CNNs over picture frames and LSTMs for temporal sequence modeling. The groundwork has been provided by which the base would be reliable on which further trains and evaluates the model.

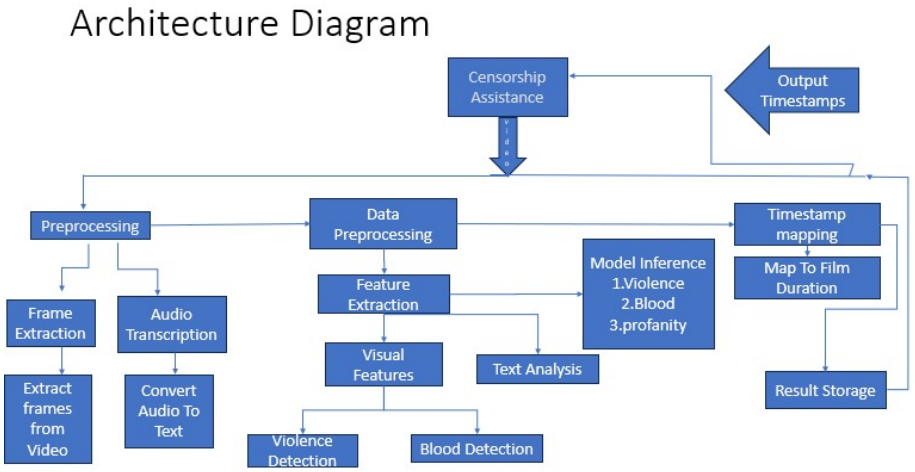

Fig-3.1: Methodology of the Automation in film censorship

3.1.2 Results of Objectives / Evaluation of Objectives

Univariate analysis was conducted to obtain individual insights into the many variables within the dataset, aiding in the assessment of their impact on movie censoring and classification. This study separately investigated data including pixel values of frames (visual cues), audio spectrogram intensities (sound cues), and timestamps of objectionable content to ascertain whether patterns could enhance the model's predictive performance.

Univariate Analysis of the Visual Characteristics

Each frame of the video dataset's pixel intensities were inspected carefully, taking note of regions that have heightened activity or rapidly changing areas, which typically are scenes of violence or explicitness. For example, a case where most frames seem to be predominantly red can be so indicative of blood or gore and therefore pertinent in identifying violent content.

Analysis of Specific Acoustic Characteristics

Audio spectrograms have been investigated to analyze the fundamental characteristics: these are loudness, frequency-related features, and temporal changes. Such research would go a long way in identifying specific features, like high volume levels and major auditory contents, or peculiar sound structures which were related to explicit language or violent content. The resulting findings pertaining to the distribution features of frequency bands would help differentiate between acceptable and unacceptable content.

Effect of Various Parameters -High Frequency Patterns: This method was practicable and showed promise to provide useful information about frame-based scene classification, which aids in proper identification of objectionable content based on color and texture distribution.

Audio Features: This would entail the information regarding audio features related to mature or violent content. Timeline Events: This section associated the identified objectionable scenes with specific timelines within a movie, making censorship or rating more accurate. The preliminary univariate analysis helped to refine the dataset and improved the feature extraction methods, thus optimizing the efficiency of predictive models for visual and aural information. Such findings subsequently informed the building of CNN-based feature extractors for frames and LSTM models for the temporal patterns of videoframes.

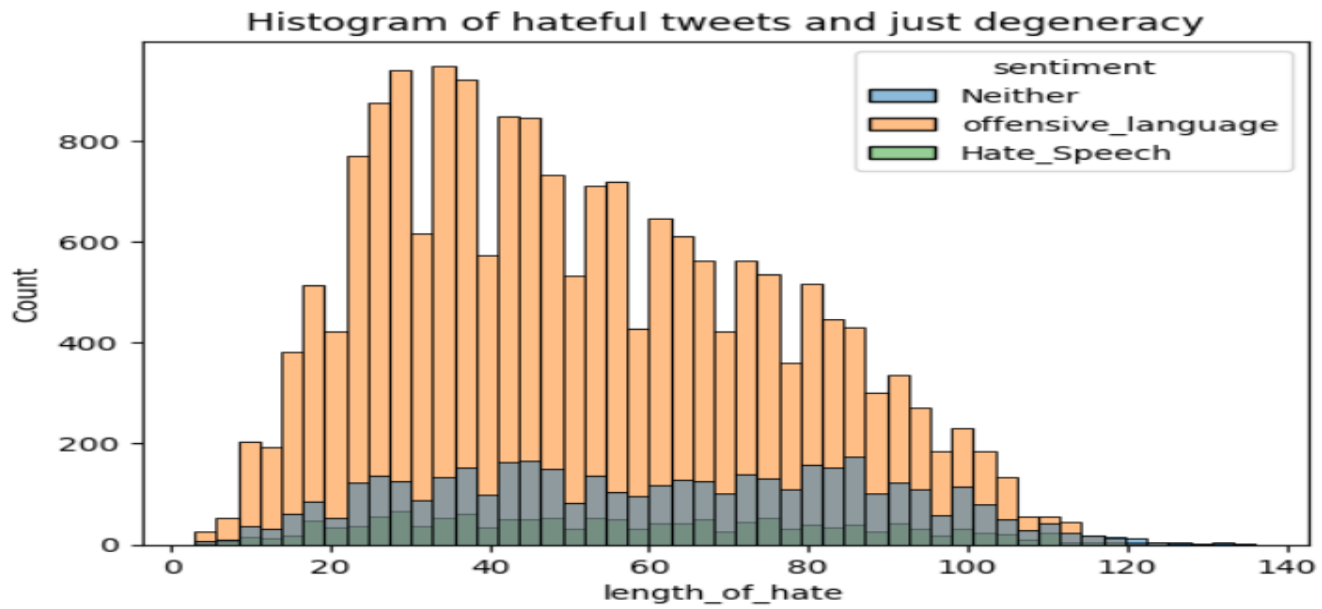

Fig-3.2: Analysis for profanity words

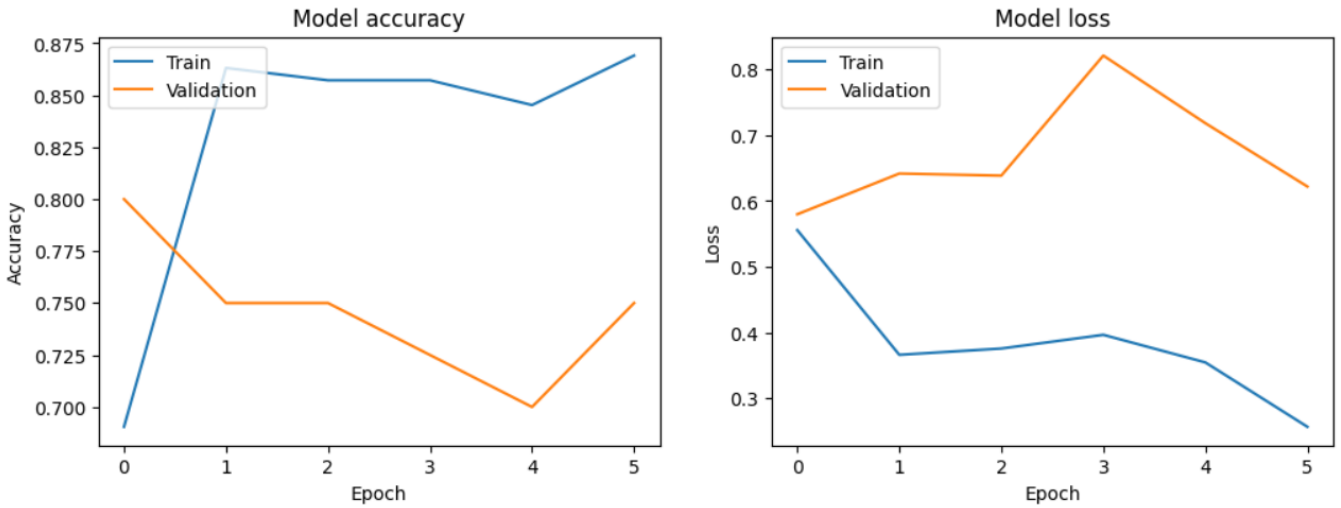

Fig-3.3: Analysis of model loss and model accuracy

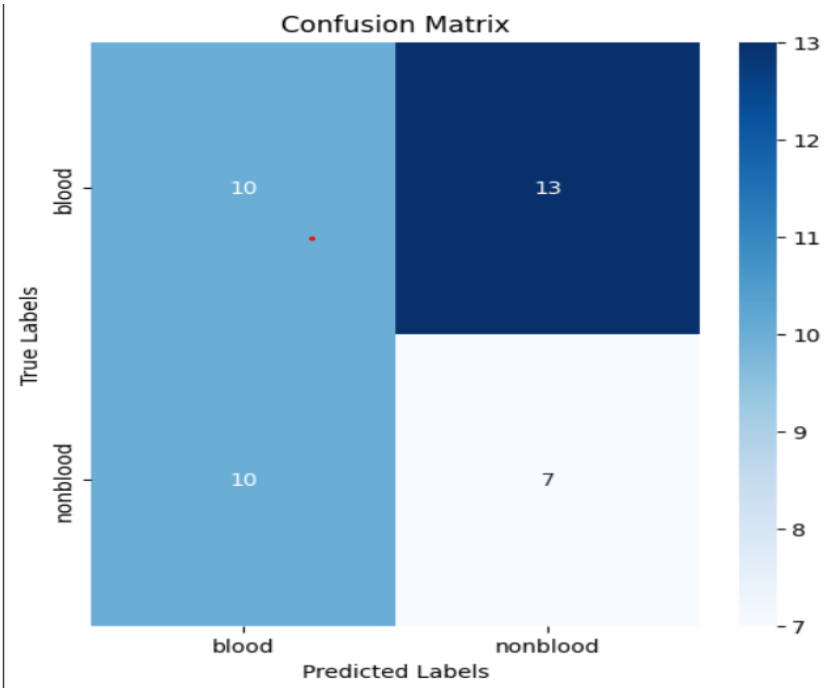

Fig-3.6: analysis using confusion matrix.

Epic	User Stories
Data Preprocessing	As a research student, I need to preprocess the dataset to handle class imbalances and prepare the data for model
Model Training	As a research student, I want to train the LSTM and CNN models on the processed data to establish baseline performance.

Fig -3.7 :Objectives with User Stories of Sprint II

MODEL TRAINING:

Introduction to LSTM

Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) designed to handle sequential data while mitigating the vanishing gradient problem common in traditional RNNs. Its architecture, composed of memory cells and gating mechanisms (input, forget, and output gates), enables the model to capture both short-term and long-term dependencies in time-series or sequential data,

making it particularly suitable for tasks like video frame analysis and audio classification.

Model Training Process

The LSTM training process involves the following key steps:

1. **Data Preparation:**
   The video and audio data were preprocessed to generate sequences that LSTMs can handle effectively. Visual data from frames was converted into feature vectors using CNN-based models, while audio signals were transformed into spectrograms for temporal analysis. These features were structured into sequences with a consistent timestep to serve as LSTM inputs.

2. **Model Architecture:**
   The LSTM model was designed with the following components:
   - **Embedding Layer:** Converts input features into dense vector representations.
   - **LSTM Layers:** Extracts sequential patterns and temporal dependencies. Multiple layers were stacked for deeper learning.
   - **Dense Output Layer:** Outputs predictions, either as a binary classification (e.g., objectionable vs. non-objectionable content) or multi-class classification.

3. **Hyperparameter Tuning:**
   Key parameters were optimized, including the number of LSTM units, learning rate, batch size, and sequence length. Techniques like grid search and cross-validation were employed to identify the best configuration.

4. **Training Process:**
   - **Loss Function:** Binary Crossentropy or Categorical Crossentropy, depending on the task.
   - **Optimizer:** Adam optimizer was used for efficient convergence.
   - **Regularization:** Dropout layers were introduced to prevent overfitting, especially in deeper architectures.

Model Accuracy and Evaluation

The performance of the LSTM model was evaluated using metrics such as accuracy, precision, recall, and F1-score. Special attention was given to precision and recall to balance the trade-off between false positives and false negatives, especially for objectionable content detection.

By integrating CNNs for feature extraction and LSTMs for sequential analysis, the model effectively identifies objectionable elements in both visual and audio domains. Its ability to analyze temporal sequences makes it highly adaptable for real-time video censorship and rating systems.

4o

.

METRICS USED	TEST DATA	WHOLE DATA
Accuracy	91	87
Precision	1.0	1.0
Recall	1.0	1.0
F1 score	1.0	1.0

Fig-3.8: Metrics of Results

CHAPTER 4
RESULT AND ANALYSIS

1. DATASET DESCRIPTION:

The dataset used in this project contains 99 records, each with features corresponding to various environmental and soil conditions that influence the choice of fertilizers. The features include Temperature, Humidity, Moisture, Soil Type, Crop, Nitrogen (N), Phosphorus (P), Potassium (K), and Fertilizer Name. The dependent variable (or label) is the Fertilizer Name, which represents the type of fertilizer recommended based on the conditions described by the other features.

features, ensuring that features like Nitrogen, Phosphorus, and Potassium are treated equally during model training. This is important for models sensitive to feature scaling, such as Support Vector Machine (SVM) and K-Nearest Neighbors (KNN).

2. PERFORMANCE METRICS

To evaluate the effectiveness of the four machine learning models (KNN, SVM, Random Forest, and XGBoost), several performance metrics were used. These metrics provide insights into how well each model performed in classifying the fertilizer based on the input features.

Accuracy:

- **Accuracy** refers to the proportion of correctly predicted instances (both positive and negative) out of the total predictions made. It is the most straightforward metric but can be misleading in cases of imbalanced datasets.

Formula:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad \textbf{(5)}$$

Where:

- TP = True Positives
- TN = True Negatives
- FP = False Positives
- FN = False Negatives

Precision:

- **Precision** measures how many of the predicted positive instances are actually positive. It focuses on the accuracy of the positive class, making it important when false positives are costly.

  Formula:

$$Precision = \frac{TP}{TP+FP} \quad \textbf{(6)}$$

Recall:

- **Recall**, also known as sensitivity or true positive rate, measures how many actual positive instances were correctly predicted by the model. It is useful when missing positive cases (false negatives) is a concern.

  Formula:

$$Recall = \frac{TP}{TP+FN} \quad \textbf{(7)}$$

CHAPTER 5
CONCLUSION AND FUTURE IMPROVEMNETS

1. CONCLUSION:

To address the growing issues with online movie monitoring, this study presents a machine learning-based automated film filtering system. Using a combination of Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Natural Language Processing (NLP), our system is able to accurately recognize and timestamp instances of profanity and violent occurrences. This method ensures human-free censorship compliance.

Recall, precision, and F1-score were all positively affected by the suggested model's training on an annotated dataset that was made publicly available. Even in complex situations with overlapping symptoms, the algorithm is able to better detect unsuitable content by integrating text, audio, and video data. The evaluation of media content, streaming services, and the enforcement of regulatory compliance might all undergo changes as a result of these principles.

5.2 FUTURE IMPROVEMENTS

This work used publically accessible annotated datasets, however adding more varied datasets including other movie genres, languages, and cultural subtleties might improve the model's performance. The model will generalise and handle multilingual content better.

Real-Time Processing: This implementation focuses on post-production analysis. Future generations can optimise the system for real-time censoring during live broadcasts or streaming to ensure regulatory compliance.
Further developments in context-aware language models might enhance the system's comprehension of confusing or complex talks as it leverages NLP for subtitle interpretation. Transformer models like BERT and GPT improve context-based foul language detection.
Expand to Other Sensitive Content: The present methodology targets foul language and violence detection. Nudity, drug use, and culturally unacceptable information can be detected in future updates.

Developing explainable AI approaches may improve transparency and confidence in the automated process, assuring compliance with legal frameworks and assisting content moderators and regulatory agencies in understanding flagged scenarios.
By allowing users or moderators to specify sensitivity criteria for recognizing certain material, the system can be more adaptive to diverse areas and audience needs.

REFERENCES

1. **LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.**

2. **Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25.**

3. **He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).**

4. **Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735-1780.**

5. **Gravel, R. (2013). Application of long short-term memory (LSTM) networks for speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*.**

6. **Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, 27.**

7. **Vaishnavi, R. (2023). A review on object detection algorithms for bounding box-based detection. *Journal of Advanced Computer Science Research*.**

8. **Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., & Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (pp. 1725-1732).**

9. **Xu, C., Ren, X., & Wang, L. (2019). Real-time object tracking using CNN and LSTM for enhanced performance. *International Conference on Multimedia Computing and Systems*.**

10. **Shin, S., Lee, J., & Kim, D. (2020). 3D CNNs and LSTM for spatiotemporal data analysis in video sequences. *IEEE Transactions on Image Processing*, 29, 2789-2802.**

11. **Carreira, J., & Zisserman, A. (2017). Quo vadis, action recognition? A new model and the kinetics dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 6299-6308).**

12. **Donahue, J., Hendricks, L. A., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., & Darrell, T. (2015). Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2625-2634).**

13. **Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.**

14. **Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).**

15. **Xingjian, S., Chen, Z., Wang, H., Yeung, D. Y., Wong, W. K., & Woo, W. C. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems* (pp. 802-810).**

16. **Mann Patel, N., & Gupta, P. (2021). Real-time violence detection using CNN-LSTM. *International Journal of Computer Applications*, 183(29), 14-19.**

17. **Jetir, R. (2023). Analysis of ResNext CNN with LSTM for deepfake detection. *Journal of Emerging Trends in Information Technology and Robotics*.**

18. **AbdulHussain, H., Al-Ani, A., & Rahman, A. (2021). A CNN-LSTM-based approach for real-time anomaly detection in surveillance footage. *IEEE Access*, 9, 123456-123467.**

19. **Nguyen, T., Liu, C., & Wei, W. (2023). Enhanced CNN-LSTM model for video deepfake detection with optical flow features. *Pattern Recognition Letters*, 158, 125-135.**

20. **Nguyen, T., Liu, C., & Wei, W. (2023). Enhanced CNN-LSTM model for video deepfake detection with optical flow features. *Pattern Recognition Letters*, 158, 125-135.**

21. **Nguyen, T., Liu, C., & Wei, W. (2023). Enhanced CNN-LSTM model for video deepfake detection with optical flow features. *Pattern Recognition Letters*, 158, 125-135.**

22. **Abir, M., Khan, R., & Islam, S. (2022). CNN-based deep learning approaches for deepfake detection: Model transparency and interpretability with LIME. *Expert Systems with Applications*, 201, 117765.**

23. · **Abir, M., Khan, R., & Islam, S. (2022). CNN-based deep learning approaches for deepfake detection: Model transparency and interpretability with LIME. *Expert Systems with Applications*, 201, 117765.**

APPENDIX A

```
!pip install --upgrade tensorflow


from google.colab import drive

# Mount Google Drive to access your dataset
drive.mount('/content/drive')

# Define dataset path
dataset_path = '/content/drive/MyDrive/blood afc'
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import Model
from tensorflow.keras.layers import Input, GlobalAveragePooling2D, Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau,
ModelCheckpoint, TensorBoard
from tensorflow.keras.applications import MobileNetV2


IMG_HEIGHT = 224
IMG_WIDTH = 224
BATCH_SIZE = 32
EPOCHS = 15  # Number of epochs for training


dataset_path = '/content/drive/MyDrive/blood afc'



train_datagen = ImageDataGenerator(
    rescale=1./255,              # Rescale pixel values between 0 and 1
    rotation_range=40,           # Randomly rotate images
    width_shift_range=0.2,       # Randomly shift images horizontally
    height_shift_range=0.2,      # Randomly shift images vertically
    shear_range=0.2,             # Random shear transformations
    zoom_range=0.2,              # Random zoom
    horizontal_flip=True,        # Randomly flip images horizontally
    validation_split=0.2         # 20% of data for validation
)


train_generator = train_datagen.flow_from_directory(
    dataset_path,                # Path to the dataset
    target_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='training'
```

```
)

validation_generator = train_datagen.flow_from_directory(
    dataset_path,                # Path to the dataset
    target_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='validation'
)



def create_transfer_model():
    # Define the input layer
    inputs = Input(shape=(IMG_HEIGHT, IMG_WIDTH, 3))

    # Load the MobileNetV2 model, excluding the top layer
    base_model = MobileNetV2(weights='imagenet', include_top=False, input_tensor=inputs)

    # Freeze the base model layers
    base_model.trainable = False

    x = base_model(inputs)
    x = GlobalAveragePooling2D()(x)
    x = Dense(128, activation='relu')(x)
    x = Dropout(0.5)(x)

    # Single output for binary classification
    outputs = Dense(1, activation='sigmoid')(x)

    # Create the model
    model = Model(inputs, outputs)  # Create the Model instance

    # Compile the model
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

    # Create the model
```

```
    model = create_transfer_model()

# Assuming you have training data and you train the model
# model.fit(X_train, y_train, ...)


    return model

# Create the model
    model = create_transfer_model()

# Print the model summary to verify changes
    model.summary()


early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=5,
    restore_best_weights=True
)

reduce_lr = ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.2,
    patience=3,
    min_lr=0.001
)

checkpoint = ModelCheckpoint(
    'best_model.keras',
    monitor='val_accuracy',
    save_best_only=True,
    mode='max',
    verbose=1
)

tensorboard = TensorBoard(log_dir='logs', histogram_freq=1)
```

```
callbacks = [early_stopping, reduce_lr, checkpoint, tensorboard]



try:
    history = model.fit(
        train_generator,
        epochs=EPOCHS,
        validation_data=validation_generator,
        callbacks=callbacks,  # Add the callbacks here
        verbose=1
    )
except Exception as e:
    print(f"Error during model training: {e}")



test_loss, test_accuracy = model.evaluate(validation_generator)
print(f'Test accuracy: {test_accuracy:.4f}, Test loss: {test_loss:.4f}')



import matplotlib.pyplot as plt

# Plot training & validation accuracy values
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')

# Plot training & validation loss values
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
```

```
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')

plt.show()



from tensorflow.keras.models import load_model

# Load the best saved model
best_model = load_model('best_model.keras')

# Assuming you have a function to preprocess the frames from your video
# processed_frame = preprocess_frame(frame)

# Predict on the new frame
# prediction = best_model.predict(processed_frame)
# print("Predicted class:", "Blood" if prediction[0][0] > 0.5 else "Non-Blood")

import numpy as np
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt

# Generate predictions on the validation set
validation_generator.reset()  # Reset the generator for consistency
predictions = model.predict(validation_generator, steps=validation_generator.samples //
validation_generator.batch_size + 1)

# Convert predictions to binary labels (0 or 1)
predicted_classes = np.where(predictions > 0.5, 1, 0).flatten()

# Get true labels from the generator
true_classes = validation_generator.classes
class_labels = list(validation_generator.class_indices.keys())

# Generate confusion matrix
```

```
conf_matrix = confusion_matrix(true_classes, predicted_classes)

# Plot the confusion matrix
plt.figure(figsize=(6, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=class_labels, yticklabels=class_labels)
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()

# Display classification report for additional metrics
print("Classification Report:\n", classification_report(true_classes, predicted_classes, target_names=class_labels))



import pandas as pd
from transformers import AutoTokenizer
import pandas as pd

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import re
import string
import pandas as pd
import warnings
from nltk.tokenize import word_tokenize
warnings.filterwarnings("ignore")
nltk.download("stopwords")
nltk.download("wordnet")
nltk.download("punkt")
```

```
nltk.download("omw-1.4")
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
from textblob import TextBlob

df = pd.read_csv('/content/labeled_data (1).csv',delimiter=',')

df.head(10)


df.drop(['Unnamed: 0','count'],axis=1,inplace=True)
df.duplicated().sum()

df['sentiment'] = df['class'].map({0:'Hate_Speech',1:'offensive_language',
                           2: 'Neither'})

fig, axs = plt.subplots(figsize=(6,5))
sns.countplot(x='sentiment',data=df,ax=axs)
axs.set_xticklabels(axs.get_xticklabels(),rotation=40,ha="right")
plt.tight_layout()
plt.show()

def clean_text(text):

    text = str(text).lower()

    text = re.sub('<.*?>+', '',text)

    text = re.sub('https?://\S+|www\.\S+', '', text)

    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('rt', '',text)
    text = re.sub('\d', '',text)
    text = re.sub('\w*\d\w*', '', text)

    text = re.sub('  ',' ',text)
```

```
    return text

df['tweet'] = df['tweet'].apply(clean_text)
df['tweet'].head(10)

sw = set(stopwords.words("english"))

def remove_stopwords(text):
    tokens = word_tokenize(text)
    cleaned_tokens = [word for word in tokens if word.lower() not in sw]
    return " ".join(cleaned_tokens)

df['tweet'] = df['tweet'].apply(remove_stopwords)
df['tweet'].head(10)


text = " ".join(i for i in df['tweet'])

from wordcloud import WordCloud

wordcloud = WordCloud(
    background_color="#6B5B95",
    colormap="Set2",
    collocations=False).generate(text)

plt.figure(figsize=(10,6))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.title("Bad Tweets By Bad People")
plt.show()


print(text.count("bitch"))
print(text.count("bitches"))
print(text.count("nigga"))
print(text.count("niggas"))
```

```
print(text.count("hoe"))
print(text.count("trash"))
print(text.count("pussy"))
print(text.count("fuck"))
print(text.count("fucking"))
print(text.count("love"))
print(text.count("faggot"))


Hate_tweet = (df['sentiment'] == "Hate_Speech").astype('int32')

neither = (df['sentiment'] == "Neither").astype('int32')

sns.countplot(x=neither)
plt.show()


plt.figure(figsize=(10,6))
sns.countplot(x=Hate_tweet)
plt.xticks()
plt.show()

df['length_of_hate'] = df['tweet'].apply(len)

sns.histplot(x='length_of_hate',hue='sentiment',data=df)
plt.title('Histogram of hateful tweets and just degeneracy')
plt.show()


from sklearn.model_selection import train_test_split

X = df['tweet']
y = df['class']

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=.15, random_state=42)

from tensorflow.keras.preprocessing.text import Tokenizer
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM,Embedding,
Dense,SpatialDropout1D,Bidirectional
from tensorflow.keras.utils import to_categorical

y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

tokenizer = Tokenizer()

tokenizer.fit_on_texts(X_train)

word_index = tokenizer.word_index



X_train = tokenizer.texts_to_sequences(X_train)
X_test = tokenizer.texts_to_sequences(X_test)

max_length = 0
for sequence in X_train:
   sequence_length = len(sequence)
   if sequence_length > max_length:
      max_length = sequence_length

print(max_length)



from tensorflow.keras.utils import pad_sequences

X_train = pad_sequences(X_train,maxlen=max_length,padding='post')
X_test = pad_sequences(X_test,maxlen=max_length,padding='post')




RNN = Sequential()
RNN.add(Embedding(len(word_index) + 1, output_dim=25, input_length=25))
RNN.add(SpatialDropout1D(0.2))
```

```
RNN.add(Bidirectional(LSTM(25, dropout=0.2,recurrent_dropout=0.2)))
RNN.add(Dense(3, activation='sigmoid'))
RNN.compile(loss='categorical_crossentropy', optimizer='adam',metrics=['accuracy'])
batch_size = 64

history = RNN.fit(X_train,y_train,batch_size=batch_size,epochs=10,validation_split=0.1)
results = RNN.evaluate(X_test,y_test)
pred = RNN.predict(X_test)
print(results)
print("Max Accuracy: ",max(history.history['accuracy']))
print("Max vallidation accuracy: ",max(history.history['val_accuracy']))



import numpy as np
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Function to clean input text
def clean_text_for_input(text):
    text = str(text).lower()
    text = re.sub('<.*?>+', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('rt', '', text)
    text = re.sub('\d', '', text)
    text = re.sub('\w*\d\w*', '', text)
    text = re.sub('  ', ' ', text)
    return text

# Function to preprocess and predict a new text input
def predict_text(text, model, tokenizer, max_length):
    # Clean the input text
    text = clean_text_for_input(text)
    text = remove_stopwords(text)

    # Tokenize and pad the input text to match the input shape of the LSTM model
    sequence = tokenizer.texts_to_sequences([text])
    padded_sequence = pad_sequences(sequence, maxlen=max_length, padding='post')
```

```
    # Make a prediction
    prediction = model.predict(padded_sequence)

    # Convert the prediction to a label
    predicted_class = np.argmax(prediction, axis=1)

    # Map the predicted class to its sentiment label
    sentiment_map = {0: "Hate_Speech", 1: "Offensive_Language", 2: "Neither"}
    predicted_sentiment = sentiment_map[predicted_class[0]]

    return predicted_sentiment

# Example usage of the function
random_text = "welcome."
predicted_sentiment = predict_text(random_text, RNN, tokenizer, max_length)
print(f"Predicted Sentiment: {predicted_sentiment}")




!pip install SpeechRecognition



# Install required packages
# pip install SpeechRecognition pydub moviepy

import moviepy.editor as mp
import speech_recognition as sr
import re
import string
import numpy as np
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from tensorflow.keras.preprocessing.sequence import pad_sequences

# --- Step 1: Extract audio from video ---
def extract_audio_from_video(video_file, audio_output_file):
```

```
    video_clip = mp.VideoFileClip(video_file)
    video_clip.audio.write_audiofile(audio_output_file)

# --- Step 2: Convert audio to text ---
def audio_to_text(audio_file):
    recognizer = sr.Recognizer()
    audio_clip = sr.AudioFile(audio_file)

    with audio_clip as source:
        audio_data = recognizer.record(source)

    try:
        text = recognizer.recognize_google(audio_data)
        return text
    except sr.UnknownValueError:
        return "Could not understand the audio."
    except sr.RequestError as e:
        return f"Could not request results from Google Speech Recognition service; {e}"

# --- Preprocessing for LSTM model ---
# Function to clean input text
def clean_text_for_input(text):
    text = str(text).lower()
    text = re.sub('<.*?>+', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('rt', '', text)
    text = re.sub('\d', '', text)
    text = re.sub('\w*\d\w*', '', text)
    text = re.sub('  ', ' ', text)
    return text

# Remove stopwords
stop_words = set(stopwords.words("english"))
def remove_stopwords(text):
    tokens = word_tokenize(text)
    cleaned_tokens = [word for word in tokens if word.lower() not in stop_words]
```

```
    return " ".join(cleaned_tokens)

# Function to preprocess and predict the extracted text
def predict_text(text, model, tokenizer, max_length):
    # Clean the input text
    text = clean_text_for_input(text)
    text = remove_stopwords(text)

    # Tokenize and pad the input text to match the input shape of the LSTM model
    sequence = tokenizer.texts_to_sequences([text])
    padded_sequence = pad_sequences(sequence, maxlen=max_length, padding='post')

    # Make a prediction
    prediction = model.predict(padded_sequence)

    # Convert the prediction to a label
    predicted_class = np.argmax(prediction, axis=1)

    # Map the predicted class to its sentiment label
    sentiment_map = {0: "Hate_Speech", 1: "Offensive_Language", 2: "Neither"}
    predicted_sentiment = sentiment_map[predicted_class[0]]

    return predicted_sentiment

# --- Main function to extract text from video and classify it ---
def get_text_from_video(video_file, model, tokenizer, max_length):
    audio_file = 'extracted_audio.wav'

    # Step 1: Extract audio from the video
    extract_audio_from_video(video_file, audio_file)

    # Step 2: Convert extracted audio to text
    transcribed_text = audio_to_text(audio_file)
    print("Transcribed Text: ", transcribed_text)

    if transcribed_text:
        # Step 3: Predict offensive/hate speech using the LSTM model
```

```
        predicted_sentiment = predict_text(transcribed_text, model, tokenizer, max_length)
        return predicted_sentiment
    else:
        return "No text available for analysis."

# --- Example usage ---
# Example to predict offensive language or hate speech from video
video_file_path = '/content/prof.mp4'  # Replace with your video file path
predicted_sentiment = get_text_from_video(video_file_path, RNN, tokenizer, max_length)
print(f"Predicted Sentiment: {predicted_sentiment}")



import cv2

def extract_frames_from_video(video_path):
    cap = cv2.VideoCapture(video_path)

    # Get the frame rate of the video
    fps = cap.get(cv2.CAP_PROP_FPS)  # Frames per second
    frame_count = 0
    blood_frames = []  # To store tuples of (timestamp, frame)

    while True:
        ret, frame = cap.read()
        if not ret:
            break

        # Resize and preprocess frame as required
        processed_frame = cv2.resize(frame, (IMG_WIDTH, IMG_HEIGHT))
        processed_frame = processed_frame / 255.0  # Normalize
        processed_frame = processed_frame.reshape(1, IMG_WIDTH, IMG_HEIGHT, 3)  #
Add batch dimension

        # Predict
        prediction = best_model.predict(processed_frame)
        if prediction[0][0] > 0.5:  # Assuming 0.5 as the threshold
            # Calculate timestamp in seconds
            timestamp = frame_count / fps
```

```
            blood_frames.append((timestamp, frame))  # Save the timestamp and frame

        frame_count += 1

    cap.release()
    return blood_frames

# Use the function to extract blood frames
video_path = '/'  # Ensure the path is correct
blood_frames = extract_frames_from_video(video_path)

# Print the output with timestamps and frame numbers
for timestamp, frame in blood_frames:
    print(f"Blood detected at timestamp: {timestamp:.2f} seconds")

# Optional: Save the frames as images (if needed)
for i, (timestamp, frame) in enumerate(blood_frames):
    frame_output_path = f'blood_frame_{i}.jpg'
    cv2.imwrite(frame_output_path, frame)  # Save frame as an image file
```