



SafeView – Automated Content Moderation

Submitted In Partial Fulfillment of Requirements

For the Degree Of

Third Year

Computer Engineering

By

Harikrishnan Gopal

&

Aditya Satish Raut

Roll Numbers:

16010122284

16010122288

Guide

Dr. Prasanna Jaichand Shete

Somaiya Vidyavihar University

Vidyavihar, Mumbai - 400 077

Academic Year 2024-25

Certificate

This is to certify that the TY Mini Project report entitled **SafeView – Automated Content Moderation** submitted by Harikrishnan Gopal (16010122284), and Aditya Satish Raut (16010122288), at the end of semester VI of TY B. Tech is a bona fide record for partial fulfillment of requirements for the degree in **Computer Engineering** of Somaiya Vidyavihar University

Guide

Examiner

Date: 24th April 2025

Place: Mumbai-77

DECLARATION

I declare that the written Mini Project report submission represents the work done based on my and / or others' ideas with adequately cited and referenced the original source. I also declare that I have adhered to all principles of academic honesty and integrity as per norms of the Somaiya Vidyavihar University. I have not misinterpreted, fabricated, or falsified any idea/data/fact/source/original work/matter in my submission.

I understand that any violation of the above will be cause for disciplinary action by the university and may evoke the penal action from the sources which have not been properly cited or from whom proper permission is not sought.

Roll NO: 16010122284	Roll NO: 16010122288
Student Name: Harikrishnan Gopal	Student Name: Aditya Satish Raut
Signature:	Signature:

Date: 24th April 2025

Place: Mumbai-77

Abstract

The SafeView project presents a real-time, browser-based solution for NSFW content classification aimed at enhancing user safety on social media platforms by detecting and filtering implicit nudity and suggestive content. Leveraging the MobileNetV3-Small model, fine-tuned on the LSPD dataset, this project addresses the limitations of traditional content moderation systems that primarily focus on overtly explicit material, often missing subtler forms of inappropriate content.

Implemented as a Chrome browser extension, SafeView utilizes TensorFlow.js for in-browser inference, enabling immediate classification of images and videos as NSFW or safe without relying on server-side processing. This design ensures privacy and efficiency, minimizing content processing latency. The performance of five distinct fine-tuning approaches was evaluated, with Full Backbone Fine-Tuning with Discriminative Learning Rates emerging as the most effective strategy, achieving the highest accuracy and balanced precision, recall, and F1-scores.

While the extension performs well across various content types, some challenges, such as false negatives and true negatives, were observed. These errors are primarily related to edge cases, such as low-resolution images or subtle NSFW content, and provide opportunities for further model refinement.

The project sets the stage for future advancements, including improving classification accuracy, optimizing latency, and expanding browser compatibility. The SafeView extension provides a robust and scalable solution for real-time, client-side content moderation, offering significant improvements in online safety, especially for vulnerable users.

Table of Content:

Chapter	Subchapter	Page No.
Chapter 1: Introduction	1.1 Introduction	7
	1.2 Motivation	
	1.3 Scope	
	1.4 Objective	
	1.5 Organization of Report	
Chapter 2: Literature Survey	2.1 Introduction	11
	2.2 Review of Existing Literature	
	2.3 Related work	
	2.4 Research Gaps and Challenges	
Chapter 3: Design Document and Project Plan	3.1 System Architecture	22
	3.2 Module Descriptions	
	3.3 Data Flow Design & Database Design	
	3.4 UI/UX Design	
	3.5 Risk Analysis & Implementation Plan	
Chapter 4: Testing	4.1 Introduction	35
	4.2 Functional Test Cases	
	4.3 Data Validation & Navigation	
	4.4 Result Verification	
Chapter 5: Prototype and Implementation	5.1 Introduction	41
	5.2 Module Description	
	5.3 Integration	
	5.4 Dataset & Sample Data (if applicable)	
	5.5 Implementation Details	
	5.6 Screenshots of Working Modules	
	5.7 GitHub Link and Contribution Summary	
Chapter 6: Results and Discussion	6.1 Performance Evaluation	62
	6.2 Comparative Analysis	
	6.3 Key Observations	
Chapter 7: Conclusion and Future Work	7.1 Conclusion	70
	7.2 Future Works	
Bibliography		73
Acknowledgements		74

List of Figures:

Figure	Page No.
Figure 1 - SafeView System Architecture	23
Figure 2 -SafeView System User Flow Diagram	27
Figure 3 - SafeView Sequence Diagram	28
Figure 4 - SafeView Activity Diagram	29
Figure 5 - LSPD Dataset Sample	49
Figure 6 - Extension UI	58
Figure 7 - HPC Environment	59
Figure 8 - Dataset Loading	59
Figure 9 - MobileNet V3 Model Architecture	60
Figure 10 - Running Epochs	60
Figure 11 - Saved Model in h5 Format	61
Figure 12 - Conversion of Model to TensorFlow.js	61
Figure 13, 15, 17, 19, 21 – Classification Report	63 – 67
Figure 14, 16, 18, 20, 22 – Confusion Matrix	63 - 67

List of Tables:

Table	Page No.
Table 1 - Literature Survey Table	15
Table 2 – Team Roles	32
Table 3 to 11 - Test cases TC-001 to TC-009	35 - 39
Table 12 – Test Case Completion	40
Table 13 – Module Description	41
Table 14 - Custom Layers in CNN	51
Table 15 to 19 - Model Training Approaches	52 - 56
Table 20 - Model Result Comparison	62

Chapter 1

Introduction

This chapter presents a brief idea about basics of Sentiments and Opinions. It deals with the purpose of taking up the project with certain motivation, scope and objectives to fulfill it.

1.1 Introduction

In today's digital landscape, social media platforms like Instagram play an increasingly central role in shaping how users consume and share content. However, as these platforms grow in popularity, they also face significant challenges in content moderation—particularly when it comes to filtering out suggestive or borderline explicit material. While current moderation policies effectively target overtly obscene content, they often fall short in identifying and controlling implicit nudity and suggestive imagery. This gap is especially concerning given the widespread presence of underage users, for whom exposure to such content may have detrimental effects. The issue underscores a critical need to revisit and enhance content moderation strategies, ensuring that the digital environment remains both safe and appropriate for all users. By examining the underlying challenges and shortcomings of existing systems, this project contributes to a broader dialogue in engineering and technology about how innovative approaches can better align policy intentions with real-world applications, ultimately promoting a safer online ecosystem.

1.2 Motivation

The motivation behind this project is rooted in the recognition of a significant gap in current content moderation practices on social media platforms. As these platforms increasingly shape public discourse and influence social behavior, their role in protecting vulnerable audiences—particularly underage users—has become paramount. Existing systems are adept at filtering overtly explicit material, yet they frequently fail to detect or appropriately manage content that, while not explicitly obscene, carries implicit nudity or suggestive cues. This oversight presents a clear practical challenge, as such content can subtly impact impressionable users, highlighting the urgent need for more refined moderation strategies.

This challenge also resonates on an academic level. The evolving nature of digital content calls for innovative applications of artificial intelligence, computer vision, and ethical technology practices to bridge the gap between policy and practice. My interest in these areas has driven a desire to explore novel approaches that not only enhance technological capability but also promote a safer online environment. This project provides an opportunity to delve into the complexities of automated content filtering and contribute to ongoing research aimed at refining these systems.

Furthermore, the project aligns with current trends and requirements in the industry, where social media companies face mounting regulatory pressures and public scrutiny over content management practices. As the demand for sophisticated, real-time moderation solutions grows, developing a system that addresses these challenges becomes both timely and relevant. By tackling a real-world problem with academic rigor and a view towards practical application, this project aims to set a precedent for responsible and effective content moderation in the digital age.

1.3 Scope

Scope of the Project

This project is focused on developing a system aimed at addressing the challenge of moderating suggestive content—content that is not explicitly obscene but contains implicit nudity or suggestive cues—on social media platforms. The goal is to create an automated tool that detects and censors such content, thereby enhancing the safety of online environments for vulnerable audiences, particularly underage users.

Inclusions:

- **Core Functionality:** The project will cover the development of an automated content moderation module capable of analyzing visual content and identifying suggestive imagery. The focus will be on detecting borderline content that current moderation policies might miss.
- **User Interface:** A Chrome extension will be created as the primary demonstration interface, showcasing how the system can analyze and censor content in real-time as it appears in a user's feed.

- **Technological Framework:** The development will involve methodologies from artificial intelligence and computer vision, including the use of machine learning techniques (e.g., deep learning models such as convolutional neural networks) to analyze images and possibly video frames. This aspect of the project will highlight the practical application of modern AI techniques in content moderation.
- **Target Audience:** The primary end users of this project are individuals who value a safe and controlled social media experience. This includes parents, guardians, and educators who are particularly concerned about safeguarding minors from exposure to suggestive content, ensuring that online interactions remain appropriate for younger audiences. Additionally, the project benefits general users who wish to curate their digital environment, providing them with a tool to filter out borderline explicit imagery that may inadvertently appear in their feeds. By focusing on these groups, the solution empowers users to take proactive steps in managing the content they encounter, promoting a more secure and customized online experience for all.

Exclusions:

- **Direct Integration with Social Media Platforms:** The project will not include direct integration with Instagram or any other social media platform's internal systems. Instead, the focus will remain on a standalone demonstration using a browser extension to simulate content moderation.
- **Extended Moderation Capabilities:** While the system targets the detection and censorship of suggestive content, it will not extend to comprehensive moderation of all types of inappropriate content. The scope is limited to managing content with implicit nudity or suggestive elements.
- **Advanced Features and Integrations:** Features beyond the core functionality—such as multi-platform support, realtime updates, advanced model training pipelines, and extensive privacy management—are beyond the project's current scope. These aspects are recognized as important but will be constrained by time and resource limitations.
- **Regulatory and Privacy Compliance:** Although ethical considerations and user privacy are acknowledged, the project will not delve deeply into the legal or regulatory aspects of content moderation. The emphasis remains on the technical feasibility and demonstration of the detection mechanism.

1.4 Objectives

The primary aim of this project is to address the limitations in current content moderation practices by developing an automated solution that detects and manages suggestive content on social media.

The objectives are outlined as follows:

Primary Objectives:

- **Develop an Automated Detection System:** Create an AI-driven solution capable of analyzing visual content to identify borderline explicit imagery—content that implies nudity or sexual suggestiveness without being overtly explicit.
- **Enhance User Safety:** Improve the digital experience for vulnerable users, particularly minors, by ensuring that suggestive content is effectively filtered from social media feeds.

Secondary Objectives:

- **Demonstrative Interface Development:** Build a Chrome extension that serves as a proof-of-concept, showcasing the system's capability to analyze and censor suggestive content in real-time.
- **Performance Evaluation:** Establish measurable metrics—such as detection accuracy, processing speed, and false positive rates—to assess and refine the system's effectiveness and reliability.
- **Documentation and Analysis:** Record the development process, methodologies used, and challenges encountered, thereby contributing valuable insights to academic and industry discussions on automated content moderation.

Tertiary Objectives:

- **Scalability Exploration:** Investigate the potential for scaling the solution to handle larger volumes of data and to integrate with other digital platforms or social media ecosystems in future iterations.
- **Future Advancements:** Identify areas for further research and development, paving the way for more comprehensive and robust content moderation strategies that can adapt to evolving digital content trends.

Chapter 2

Literature Survey

The Objective of a literature survey is to review existing research, identify gaps, and establish a strong foundation for the study. It helps in understanding key concepts, comparing different approaches, and justifying the need for the current research by analyzing past studies.

2.1 Introduction

The explosive growth of user-generated visual content online has underscored the urgent need for automated, robust methods to detect and filter pornographic or otherwise inappropriate material. In the context of web browsing—particularly via browser extensions—real-time classification and moderation of images and videos is critical to protect vulnerable users, comply with platform policies, and reduce reliance on manual moderation. This literature survey surveys existing research on image- and video-based nudity/NSFW detection, from traditional machine-learning approaches through modern deep-learning and multimodal methods, identifying strengths, limitations, and research gaps.

2.2 Review of Existing Literature

Early SVM and Skin-Color Methods (2003–2011):

Fleck et al. pioneered nudity detection via skin-color segmentation and SVM classifiers, achieving $\approx 92\%$ accuracy on curated photo sets. While groundbreaking, these approaches struggled with false positives in benign skin-exposed contexts like sports.

Bag-of-Visual-Words and Motion Features (2005–2016):

Laptev’s space-time interest-point descriptor combined with BoVW models improved robustness to simple skin-based filters. Later, Moreira et al. introduced Temporal Robust Features to capture spatio-temporal cues in video, yielding $>93\%$ accuracy on the NPDI dataset but at the cost of heavy preprocessing.

Deep CNNs and Transfer Learning (2015–2018):

Moustafa et al. demonstrated that off-the-shelf networks like AlexNet and GoogLeNet—fine-tuned via transfer learning—could attain $\sim 94\%$ accuracy on NPDI-800. and Agastya et al. used VGG16 and ResNet variants to push classification accuracy above 95% on benchmark datasets LSPD.

Advanced Architectures and Ensemble Schemes (2019–2021):

Song and Kim proposed a multimodal stacking ensemble of VGG-16 and BiRNN to exploit both visual and auditory streams in video, achieving 95.4% true positive rate with low false negatives—key for live moderation.

compared optimizers on EfficientNet-V2, finding ADAMX-optimized V2L models reached 95.6% test accuracy on a proprietary NSFW image set.

In-Browser Inference and Client-Side Moderation (2022–2024):

Karabulut et al. designed a deep network pipeline for automatic content moderation, coupling classification with region-based obfuscation—demonstrating viability for real-time, client-side tools Automatic content model. explored GPT-based systems for text moderation, highlighting the promise and limitations of LLMs for personalizable filtering.

2.3 Related Work

1. Akyon & Temizel (2023) – “State-of-the-Art in Nudity Classification: A Comparative Analysis”

- **Summary**
Benchmarks several CNNs (MobileNetV3 small/large, Inception v3, ConvNeXT) and a ViT(B16) on three nudity datasets (AdultContent, NudeNet, LSPD). ConvNeXT-tiny achieves the highest F1~0.95; ViT suffers slow convergence.
- **Relevance**
Demonstrates that lightweight CNNs can reach >94 % F1 and highlights LSPD as a robust benchmark—exactly the dataset we fine-tune MobileNetV3 on for real-time NSFW detection.
- **Comparison**
They use offline training/evaluation; we push a MobileNetV3-Small into TensorFlow.js for in-browser inference. Both target $\geq 94\%$ accuracy, but we add a browser UI and real-time CSS blurring.
- **Critical Analysis**
Their datasets lack “web-native” images (thumbnails, memes). This domain gap motivates fine-tuning further on browser-scraped samples to avoid real-time misclassifications.

2. Phan et al. (2022) – “LSPD: A Large-Scale Pornographic Dataset for Detection and Classification”

- **Summary**
Introduces LSPD: 500 k images + 4 k videos labelled into five classes (safe, sexy, porn, hentai, drawing) plus pixel-masks for four sexual objects. Provides classification and object-detection baselines (ConvNeXT, YOLO, Mask-RCNN).
- **Relevance**
LSPD is our primary training corpus for the NSFWX MobileNetV3-Small model—its diversity (hentai, drawings) ensures broad coverage of web content.
- **Comparison**
Phan *et al.* also tackle object segmentation; we focus solely on 5-way classification and region-agnostic blurring via Grad-CAM.
- **Critical Analysis**
Though exhaustive, LSPD’s studio-grade imagery may not reflect low-res thumbnails or composite web graphics—which we’ll sample and fine-tune on in the browser environment.

3. Arora et al. (2023) – “ADAMAX-Based Optimization of EfficientNetV2 for NSFW Content Detection”

- **Summary**
Compares ADAM, ADAMAX, ADAMW, SGD, ADAGRAD on EfficientNet-V2M/L for NSFW image classification. ADAMAX on V2L yields 98.8 % train and 95.6 % test accuracy.
- **Relevance**
Informs choice of optimizer and backbone for high offline accuracy—key when fine-tuning on LSPD before TensorFlow.js conversion.
- **Comparison**
We prefer the smaller MobileNetV3-Small to meet browser footprint constraints; they optimize a much larger V2L for pure offline accuracy.
- **Critical Analysis**
Their focus on offline accuracy overlooks client-side latency—our work measures JS inference times and trades a bit of accuracy for sub-200 ms per image.

4. Song & Kim (2020) – “An Enhanced Multimodal Stacking Scheme for Online Pornographic Content Detection”

- **Summary**
Uses VGG-16 + BiRNN for visual features and a dilated-conv block for audio, stacking fusion, video, and audio classifiers. Achieves 95.4 % TPR and 4.6 % FNR, with rapid detection up to 74.6 % faster than prior schemes.
- **Relevance**
Shows how early fusion can boost recall (low false negatives)—critical for user safety in a browser extension.
- **Comparison**
Our prototype currently processes only images in real time; Song & Kim’s optical-flow and RNN approach is too heavy for in-browser use.
- **Critical Analysis**
Their motion/audio modules improve FNR but incur high compute; our plan to remain single-frame ensures sub-200 ms detection, trading some FNR for real-time practicality.

5. Karabulut et al. (2022) – “Automatic content moderation on social media”

- **Summary**
Proposes a two-stage pipeline: CNN classification + region-based obfuscation via class activation maps (CAM). Trains a noise-robust pipeline on proprietary data, achieving 90.3 % top-1 accuracy and mean occlusion ≥ 0.68 .
- **Relevance**
Direct precursor to our Grad-CAM→CSS-blur technique for browser integration, demonstrating how to automatically mask regions without extra annotations.
- **Comparison**
We adapt their CAM-based masking to run in JS and add a full settings UI, whereas they stop at proof-of-concept masking.

- Critical Analysis
Their obfuscation is effective but lacks user controls; *SafeView* adds blur-strength sliders and site whitelisting for better UX.

6. Samal et al. (2023) – “Obscene image detection using transfer learning and feature fusion”

- Summary
Generates a GAN-augmented GGOI dataset via Pix2Pix, fine-tunes multiple TL backbones (MobileNetV2, DenseNet169), and fuses their mid-/low-level features. Fusion of MobileNetV2 + DenseNet169 reaches 98.5 % accuracy and 98.5 % F1 on Pornography-2K and NPDI.
- Relevance
Demonstrates that feature-fusion of compact backbones can boost accuracy—an idea we can explore if single-frame MobileNetV3 shows gaps.
- Comparison
We opted for a single TL model to minimize conversion complexity in TensorFlow.js; Samal *et al.*’s fusion would increase model size and JS load times.
- Critical Analysis
Their GAN-based augmentation enhances accuracy offline but adds training complexity. For *SafeView*, simpler augmentation on LSPD suffices to meet in-browser constraints.

7. Yousaf & Nawaz (2022) – “A Deep Learning-Based Approach for Inappropriate Content Detection and Classification of YouTube Videos”

- Summary
Extracts frame-level features with EfficientNet-B7, feeds them into a BiLSTM+attention block, and classifies segments of cartoon videos as safe, violent, or sexual with 95.7 % accuracy on a custom YouTube cartoon clips dataset.
- Relevance
Illustrates how efficient backbones + temporal models can classify web video in the client; though *SafeView* currently handles static images, the same frame sampling + tfjs approach could extend to short clips.
- Comparison
We trade temporal aggregation (BiLSTM) for single-frame speed, since running LSTM in JS would exceed performance budgets.
- Critical Analysis
Their approach excels on well-scoped video clips but is too heavyweight for arbitrary web pages; we prioritize immediate, one-shot classification.

8. Franco et al. (2024) – “Integrating Content Moderation Systems with Large Language Models”

- **Summary**
Explores using GPT-3.5/LLAMA2 in moderation pipelines, enabling personalized content policies and user appeals. Benchmarks LLMs vs. commercial classifiers on toxicity/hate datasets.
- **Relevance**
Highlights the future potential of LLMs for on-the-fly, personalized moderation—complementary to *SafeView*’s image focus.
- **Comparison**
Our extension uses a small convnet client-side; LLMs require API calls—impractical for privacy, latency, and offline use.
- **Critical Analysis**
While LLMs offer fine nuance, they carry privacy/latency costs. A hybrid client + cloud approach could be future work once on-device image filtering is stable.

9. Gutfeter et al. (2024) – “Detecting sexually explicit content in the context of the CSAM: End-to-end classifiers and region-based networks”

- **Summary**
Compares an end-to-end CNN classifier vs. a two-step person+body-part detector on restricted CSAM data from a closed system. End-to-end CNN yields 90.2 % accuracy after augmenting with adult porn and neutral samples.
- **Relevance**
Shows that end-to-end classification can outperform complex, multi-module pipelines—justifying our single-shot MobileNetV3 approach.
- **Comparison**
They focus on CSAM specifically, in a controlled environment; *SafeView* addresses broader NSFW contexts on arbitrary web pages.
- **Critical Analysis**
Their dataset is small/closed for privacy; our public LSPD fine-tuning offers wider generalizability for web scenarios.

10. Borj et al. (2021) “Machine-Learning Approach to Detect Predatory Chat”

- **Summary**
Uses GBDT on hand-crafted linguistic features to detect grooming chats in private messaging.
- **Relevance**
Highlights dangers in text modalities—complementary to image filtering but out of scope for *SafeView*.
- **Comparison**
Our focus is client-side image/video; they require server-side text preprocessing.
- **Critical Analysis**
Text detection demands NLP pipelines; browser-extension text scanning raises privacy/complexity concerns beyond our current scope.

11. “A Machine-Learning Based Adult Content Detection Using SVM” (2020)

- **Summary**
Extracts image embeddings via MobileNetV2 and DenseNet, trains an SVM to classify adult vs. safe images. Reports ~95 % accuracy on a private dataset.
- **Relevance**
Illustrates an efficient embedding+SVM pipeline, showing that lightweight extractors+classifiers can work well.
- **Comparison**
We opt for end-to-end CNN in TensorFlow.js rather than splitting CNN+SVM—simplifying client inference.
- **Critical Analysis**
SVM adds a separate classification layer not easily ported to JS; our unified model is more practical for real-time in-browser use.

Paper Title	Methodology	Dataset Used	Observation of Proposed Methodology	Pros	Cons	Findings
A Deep Learning-Based Approach for Inappropriate Content Detection and Classification of YouTube Videos (K. Yousaf, 2022)	CNN-BiLSTM framework with EfficientNet-B7 for feature extraction and BiLSTM for video classification.	Cartoon video dataset from YouTube (111,156 video clips)	Achieved a validation accuracy of 95.66% and the highest recall score of 92.22%. Video preprocessing, feature extraction, and classification are key.	High accuracy in detecting inappropriate content in children-oriented videos. Potential for assisting video platforms in removing unsafe content.	Existing studies focused on binary classification (safe/unsafe) and lacked real-time multi-class classification for children's videos.	The EfficientNet-BiLSTM framework demonstrated superior performance compared to other models for detecting inappropriate content in children's videos.
ADAMAX-Based Optimization of Efficient Net V2 for NSFW Content Detection	Comparison of optimizers (ADAM, ADAMAX, ADAMW, SGD, ADAGRAD) on	9028 NSFW images (3009 training, 3019 validation, 3000 testing)	The ADAM optimizer outperformed others, achieving 98.80% training accuracy and	Provides insights into optimizer selection for NSFW content detection. Aims to create safer	The dataset lacked diversity in terms of ethnicities and lighting conditions, suggesting	The ADAM optimizer outperformed others in terms of accuracy, with the EfficientNet

K. J. Somaiya School of Engineering
Department of Computer Engineering

(Gaurav Raj, 2023)	EfficientNet-V2M and V2L models.		95.60% testing accuracy on EfficientNet-V2L.	environments for children and teenagers.	room for future improvement.	V2L model showing superior performance in NSFW content detection.
An Enhanced Multimodal Stacking Scheme for Online Pornographic Content Detection (K.S., 2020)	Enhanced multimodal stacking scheme using VGG-16 for visual features, bidirectional RNN for audio features, and stacking ensemble for classification.	8000 video segments (4000 hazardous, 4000 non-hazardous)	Achieved 92.33% accuracy and a lower false negative rate (4.60%) compared to other methods. Detected 95.40% of harmful content.	Superior performance with fewer false negatives. Utilizes visual and auditory features for better detection.	Difficulty in directly comparing results due to varying experimental setups and datasets.	The enhanced multimodal stacking scheme demonstrated superior performance in pornographic content detection using both visual and auditory data.
AttM-CNN: Attention and Metric Learning-Based CNN for Pornography, CSA Detection (Abhishek Gangwar, 2020)	Attention and Metric Learning-based CNN incorporating visual attention mechanisms and center loss for better feature discrimination.	NPDI Pornography-2k dataset, CSA dataset (1M adult, 5k CSA images)	Achieved 97.10% accuracy on the NPDI dataset and 92.72% for CSA classification. The attention mechanism helps focus on relevant image regions.	Improved accuracy in detecting pornography and CSA compared to forensic tools. Attention mechanisms enhance feature focus.	Performance comparison relies on results from other publications with potentially different experimental setups.	AttM-CNN achieved state-of-the-art results in pornography and CSA image detection and classification, also showing strong performance in age-group classification.
Automatic Content Moderation on Social Media (Dogus	Neural network-based classification with gradient-weighted class activation	Images scraped from hand-selected subreddits, categorized into 5	Classification accuracy ranging from 87.9% to 90.3%. Obfuscation	Cost-effective solution for content moderation, reducing human labor.	Dataset cannot be shared publicly due to the nature of the data. The quality of	The proposed content moderation pipeline combines content

K. J. Somaiya School of Engineering
Department of Computer Engineering

Karabulut, 2022)	maps for obfuscation.	classes: Explicit Nudity, Suggestive Nudity, Neutral, Sexually Explicit Artwork, Neutral Artwork.	algorithm masks inappropriate parts based on classification.	Obfuscation derived from the classification network reduces the need for separate training data.	the dataset is limited by the subreddits chosen.	classification with obfuscation, offering a scalable and cost-effective solution for social media platforms.
Detecting Sexually Explicit Content in CSAM Context (W.Gutfeter, 2024)	Classification approaches: end-to-end classifier, person-based patch classification, and body part detection.	NASK-Dyžurnet dataset (7688 samples)	Focused on data analysis and proposes classification schema. False positive results for neutral images and adult pornography.	Provides insights into CSAM data characteristics with varying levels of classification granularity.	Limited true positive detection results due to the sensitive nature of the data, focusing more on classification schema and false positives.	Proposed a multi-faceted classification schema that integrates end-to-end, person-based, and body-part-based approaches for CSAM detection.
Discovering Child Sexual Abuse Material Creators' Behaviors on the Dark Web (V.M. Ngo, 2024)	Supervised learning with ML and DL algorithms (SVM, CNN, LSTM, BERT) for text classification of CSAM-related posts.	ATLAS dataset from dark web forums (4600 labeled posts)	SVM model achieved 92.3% precision and 87.6% accuracy in CSAM text post classification. Extracted metadata about CSAM creators and victims.	High performance in precision and accuracy for CSAM text detection. Extracts valuable insights about CSAM creators and victims.	The research only focused on English posts. SVM did not achieve 100% precision. Keyword matching and basic NLP for metadata extraction might overlook some information.	Developed a high-performance CSAM detection system that extracts meaningful information about CSAM activities on the dark web.
Integrating Content Moderation	Integration of LLMs (ChatGPT,	OpenAI's Dataset (1680	LLMs showed promise for content	LLMs offer personalized and context-	Challenges in obtaining numerical	LLMs can be effectively integrated

K. J. Somaiya School of Engineering
Department of Computer Engineering

Systems with Large Language Models (M. Franco, 2024)	LLaMa 2) within content moderation systems, framed as a question-answering task based on defined rules.	samples of undesired content), Reddit Multilingual Content Moderation dataset (1.8 million comments).	moderation with varying performance across datasets and subreddits. LLMs can provide explanations for moderation decisions.	sensitive moderation. They address limitations of traditional ML-based solutions that lack contextual understanding.	confidence values from LLMs. Biases in LLM training data might influence moderation decisions.	into content moderation systems, especially for nuanced community rules and personalized moderation.
LSPD: A Large-Scale Pornographic Dataset for Detection and Classification (Dinh Duy Phan, 2022)	Introduction of a large-scale, high-quality public dataset and experimental evaluation of methods on it.	LSPD dataset (500,000 images, 4,000 videos)	Introduced a comprehensive dataset with detailed annotations for both pornography detection and sexual object detection tasks.	Provides a large, publicly available dataset for pornography detection and sexual object identification.	Limited in scope to specific types of content. Further exploration of methodologies using this dataset is required.	LSPD dataset offers a valuable resource for training and evaluating algorithms for pornography and sexual object detection.
Obscene Image Detection Using Transfer Learning and Feature Fusion (Sonali Samal, 2023)	Transfer learning from MobileNetv2 and DenseNet169 with feature fusion for image classification.	NPDI, Openimage, GGOI datasets	Achieved an average classification accuracy of 99.25%, outperforming state-of-the-art methods across datasets.	High classification accuracy with feature fusion, balancing accuracy and computational complexity.	Computational complexity is higher than simpler models like GoogleNet.	The TLMobDense model demonstrated superior accuracy in obscene image detection while maintaining a reasonable computational cost.
State-of-the-Art in Nudity Classification: A Comparative	Comparative analysis of methods in nudity classification	Various publicly and privately available datasets	Provides a structured overview of nudity classification	Offers a comprehensive comparison of methodologies and datasets	The excerpt does not provide specific pros and cons or	The paper offers a broad comparative analysis of nudity

Analysis (Fatih Cagatay Akyon, 2023)	using color, shape, and deep learning techniques.		techniques, highlighting deep learning's dominance in recent studies.	used in nudity classification.	detailed findings for individual methods.	classification techniques, shedding light on strengths and weaknesses of various approaches.
---	--	--	---	-----------------------------------	--	---

Table 1: Litreture survey comparision

2.4 Research Gaps and Challenges

1. Low Accuracy of Existing Models

- Sub-optimal performance ($< 95\%$)
 - Many published methods—particularly those trained on limited datasets (e.g. NPDI-800, Pornography 2k)—report overall classification accuracies in the low- to mid-90 percent range. While respectable, this level of performance still allows one misclassification in twenty, which is unacceptable in high-stakes contexts (e.g. protecting minors).
- Poor fine-grained discrimination
 - Models often collapse “sexy” or borderline content into either “safe” or “explicit,” leading to elevated false-positive or false-negative rates when distinguishing nuanced categories (e.g. “suggestive” vs. “explicit”).

2. Under-utilization of State-of-the-Art Image Detection Networks

- Limited transfer-learning
 - Besides a few works that borrow from ImageNet-trained CNN backbones (e.g. AlexNet, VGG, ResNet), the majority of research does not leverage modern object-detection architectures—such as Faster R-CNN, YOLOv5/YOLOv8, EfficientDet, or DETR—that can be fine-tuned for NSFW detection and yield superior feature representations.
- No multi-task or region-based adaptations
 - Sexual-object detectors (breasts, genitals, anuses) have been proposed in isolation, but few systems integrate these with whole-image classifiers to jointly localize and classify NSFW content—missing an opportunity to improve accuracy by combining regional cues with global context.

3. Reliance on Small, Non-Representative Datasets

- Dataset scale and diversity
 - Standard benchmarks (e.g. NPDI, Pornography 2k) contain only a few thousand images or videos—insufficient to capture the vast stylistic and cultural variation in real-world NSFW content.

- Lack of up-to-date, synthetic, and cross-cultural samples
 - Very few datasets include user-generated adult content from social media, GAN-synthesized imagery, or images from non-Western contexts, leading to models that overfit to outdated or biased corpora.

4. In-Browser Latency and Model Footprint

- Heavy model sizes
 - Current in-browser solutions (e.g., TensorFlow.js with MobileNetV2, EfficientNet-Lite) still weigh in at several megabytes, resulting in large download sizes and high memory usage.
- Real-time performance constraints
 - Typical frame rates plunge below 5 FPS when performing full-page scans on consumer devices, leading to poor user experience and missed detections.
- Energy and CPU overhead
 - Continuous NSFW inference on images and videos can rapidly drain battery life on mobile devices and overload CPU or GPU resources.

5. Lack of User Personalization

- Uniform thresholds and filters
 - Almost all systems apply a one-size-fits-all NSFW threshold; none allow end users to specify personally acceptable levels of “suggestiveness,” “minimal nudity” or differentiate “medical” vs. “erotic” contexts.
- No adaptive learning from user feedback
 - There is no mechanism for users to correct false positives/negatives and have the model incrementally adapt to individual preferences or community norms.

Chapter 3

Design Document and Project plan

A design document is crucial in a software project because it serves as a blueprint that outlines the architecture, components, data flow, and technical specifications of the system before implementation. Clear Vision & Planning will improve collaboration with in the team members. The

3.1 Introduction:

The purpose of this document is to outline the SafeView project's design, including its system architecture, core components, and key design principles. SafeView is an automated content moderation tool aimed at detecting and censoring suggestive, implicit nudity, or NSFW (Not Safe for Work) content, particularly on social media platforms and websites.

Expected Audience:

This document is intended for developers, system architects, and project stakeholders involved in the SafeView project, including those responsible for its deployment, maintenance, and further development.

Scope of the Project:

SafeView is focused on creating an automated tool that detects and censors suggestive content, including implicit nudity, from media found on social media platforms. The project includes building a system capable of real-time content detection and automatic filtering based on user-configurable settings.

Definitions, Acronyms, and Abbreviations:

NSFW: Not Safe for Work. This refers to content that is explicit, sexual, or otherwise inappropriate for certain audiences.

SafeView: The content moderation tool designed to automatically blur suggestive or NSFW content on web pages.

NSFWX: The fine-tuned version of the MobileNetV3-Small model used in SafeView for detecting NSFW content.

Detector: The core engine responsible for analyzing content, including NSFW and face detection.

Queue: The system that manages the processing of images and videos, ensuring that each piece of media is analyzed in sequence without overloading the system.

References:

TensorFlow.js Documentation: Official documentation on how to use TensorFlow.js for running models directly in the browser.

Web Browser Extension Development: A reference for developing browser extensions and understanding how to interact with the DOM to apply style changes dynamically, which is a key part of SafeView for applying blur effects to images/videos.

MobileNetV3 Documentation: MobileNetV3 is the model architecture used for detecting NSFW content.

3.2 System Over View

1. System Architecture:

SafeView - System Architecture

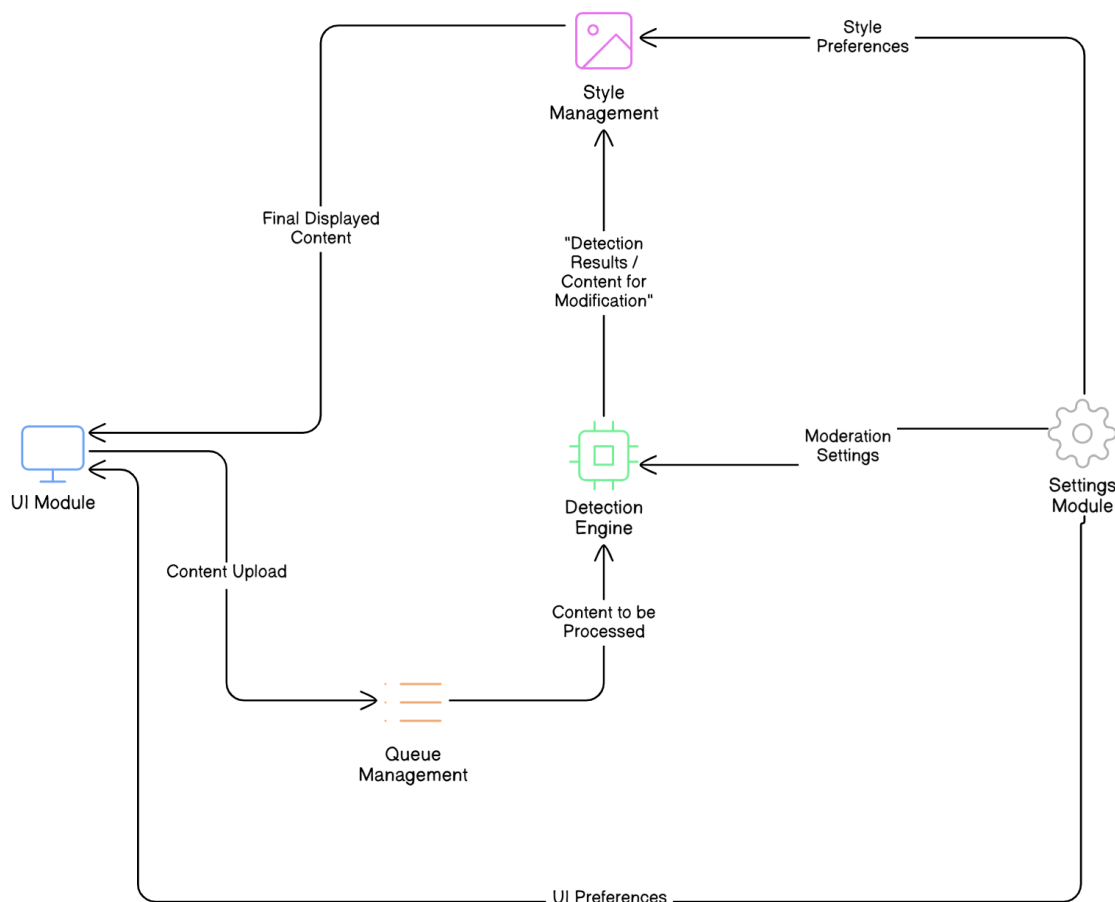


Figure 1: SafeView System Architecture

2. Design Goals

The following are the key design principles that guide the development of SafeView:

- **Scalability:**
 - Goal: To ensure the system can handle a high volume of images and videos efficiently, especially for social media platforms with millions of posts.
 - Design Consideration: Implement a queue management system that processes images/videos sequentially but can scale horizontally if needed. The use of offscreen processing helps prevent UI blocking and ensures smooth performance.
- **Security:**
 - Goal: To ensure that SafeView operates securely without compromising user privacy or web data.
 - Design Consideration: The extension operates entirely in the browser, with no personal data being sent to external servers. The system processes content locally to detect and blur inappropriate media. No private browsing data is collected or stored.
- **Performance:**
 - Goal: The system should process content in real-time, with minimal latency.
 - Design Consideration: The use of TensorFlow.js models enables the system to run directly in the browser, avoiding round-trip delays to servers. Models are optimized for performance, and media is queued for efficient processing. Background processing ensures that the browser's UI is not overwhelmed.
- **Usability:**
 - Goal: To provide an intuitive interface for users, enabling them to configure content filtering with ease.
 - Design Consideration: The UI is designed with user-friendliness in mind, allowing users to quickly toggle settings such as blur intensity, grayscale, and content detection on/off for images and videos.

3.3 Detailed Design

1. Module Description

SafeView consists of several key modules, each responsible for a specific functionality. Below is a description of these modules, their responsibilities, and interactions:

1. Model Training Module

The **NSFW Model Training** module is designed to train a machine learning model that classifies images into five NSFW categories: Drawing, Hentai, Non-Porn, Porn, and Sexy. It uses **MobileNetV3-Small**, fine-tuned with the **LSPD dataset**, and optimized for real-time inference within the **SafeView Chrome extension**.

Key Components:

- **Data Preprocessing:** The LSPD dataset is split into training, validation, and test sets, with images resized to **224x224 pixels** and normalized for model input.
- **Model Architecture:** Utilizes **MobileNetV3-Small** as the base model, with custom layers like global average pooling, dropout, and dense layers for multi-class classification.
- **Model Training:** The model is trained using **Adam optimizer**, **Categorical Cross-Entropy** loss, and accuracy as the evaluation metric.
- **Model Evaluation:** Performance is assessed on a test set with metrics like **accuracy**, **precision**, **recall**, **F1-score**, and confusion matrix.
- **Model Deployment:** The trained model is converted to the **TensorFlow.js** format for use within the SafeView extension.

2. User Interface (UI) Module:

- **Responsibility:** Provides the frontend interface for users to interact with SafeView, including settings for content moderation and feedback on content status (blurred or safe).
- **Interactions:**
 - Interacts with the Queue Management module to initiate content filtering.
 - Communicates with the Detection Engine to show detection results.
 - Allows users to configure content detection preferences (e.g., blur intensity, enabling/disabling content moderation on specific websites).

3. Queue Management Module:

- **Responsibility:** Handles the flow of media (images/videos) to be processed by the detection engine, ensuring that each piece of content is analyzed in sequence.
- **Interactions:**
 - Receives media requests from the UI and places them in the processing queue.
 - Sends media items to the Detection Engine for content analysis.
 - Communicates results back to the UI and the Style Management module for applying changes (e.g., blurring).

4. Detection Engine (NSFWX Model):

- **Responsibility:** The core engine responsible for analyzing media content (images/videos) using the NSFWX model, which detects and classifies NSFW content.
- **Interactions:**
 - Receives media from the Queue Management module.
 - Processes the content using the NSFWX model to detect NSFW or inappropriate content.
 - Sends the detection results (NSFW classification) to the Style Management module for action (blur/unblur).
 - Optionally interacts with a facial recognition model to detect faces and apply specific filtering based on user preferences.

5. Style Management Module:

- **Responsibility:** Manages the visual representation of content based on detection results. This module is responsible for applying dynamic CSS to blur or unblur content.
- **Interactions:**
 - Receives detection results from the Detection Engine.
 - Applies visual changes (e.g., blurring, grayscale) to content in the DOM based on user settings and detection results.
 - Communicates with the UI to reflect the current state of content moderation (e.g., whether content is blurred or visible).

6. Event Handling Module:

- **Responsibility:** Listens for user-triggered events and dynamically adjusts the system based on user input, such as toggling content moderation, adjusting blur intensity, or enabling/disabling features.
- **Interactions:**
 - Communicates with the UI to update settings and preferences.
 - Sends signals to the Queue Management and Detection Engine to initiate actions based on new configurations.

7. Settings Module

- Responsibility: Stores and manages user preferences for content moderation, including blur intensity, grayscale mode, and enabling/disabling features like male or female face detection.
- Interactions:
 - Receives and updates settings from the UI based on user input (e.g., enabling or disabling content filtering).
 - Stores settings in local storage (e.g., chrome.storage.sync).
 - Sends updated settings to the UI, Queue Management, and Detection Engine modules.

2. Data Flow & Components

System user flow diagram:

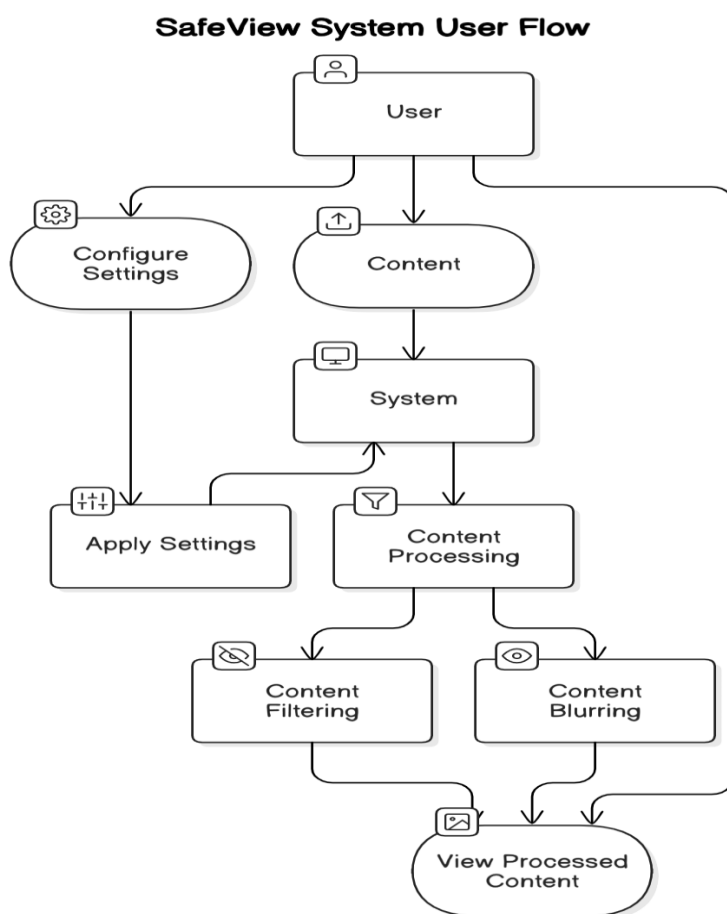


Figure 2: SafeView System user flow diagram

Sequence Diagram:

SafeView System Sequence

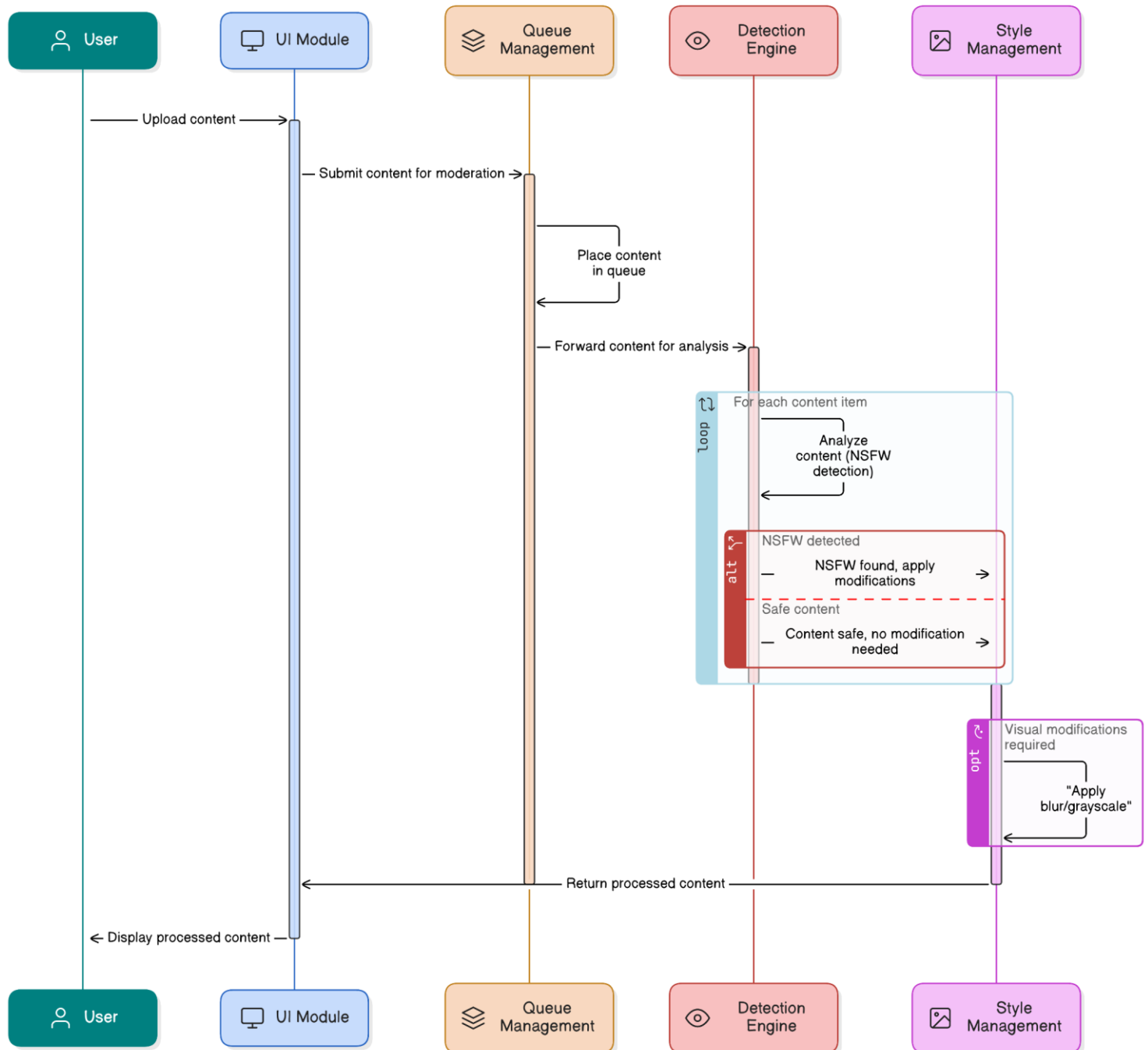


Figure 3 : SafeView Sequence Diagram

Activity Diagram:

Content Moderation Process

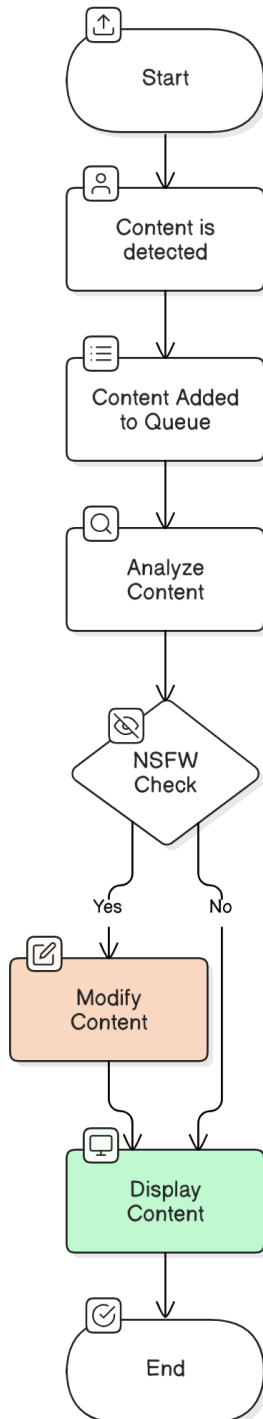


Figure 4: SafeView Activity Diagram

3. Database Design

Although **SafeView** does not require a traditional database for storing user data or media, it may store temporary metadata related to content detection and user preferences.

Metadata Table:

- Purpose: Stores metadata about the media processed by the system, such as content type, detection result, timestamp, and user preferences.

User Preferences Table:

- Purpose: Stores individual user preferences for content filtering

Model training database:

- For storing and managing your **LSPD dataset** of **500,000 images** (97 GB), we will focus on an efficient structure for the **training**, **validation**, and **test** sets that ensures easy access, scalability, and maintainability. Since the dataset is stored in a **HPC (High-Performance Computing) system storage**, the emphasis here is on how the dataset is organized and accessed for training and inference, specifically for **model training**.
- **Dataset Characteristics and Distribution**
 - **Drawing**: 50,000 images
 - **Hentai**: 50,000 images
 - **Non-Porn**: 150,000 images
 - **Porn**: 200,000 images
 - **Sexy**: 50,000 images

4. User Interface Design

The UI Design for SafeView should provide a simple and intuitive experience for the user. Below are key elements:

UI Elements:

- Toggle Switches for enabling/disabling content moderation.
- Sliders for adjusting blur intensity and grayscale filtering.
- Checkboxes for selecting which media types to blur (images, videos) and whether to blur faces based on gender.
- Live Feedback to inform the user if content is being processed or blurred.

Wireframes:

1. Main Interface:
 - A simple dashboard with options to toggle NSFW Detection on/off, adjust blur intensity, and enable/disable grayscale filtering.
 - A preview area shows images/videos with blur applied.
2. Settings Page:
 - Contains sliders for blur intensity, a checkbox for grayscale, and toggles for enabling/disabling content moderation.
 - Whitelist management allows users to exclude specific websites from content detection.

5. External Interfaces

APIs and Third-Party Services:

1. **TensorFlow.js:**
 - Used for running the **NSFWX** model directly in the browser.
 - **TensorFlow.js API:** Allows seamless integration of pre-trained models into the web application for real-time content analysis.
2. **Browser Extension APIs:**
 - **WebExtension APIs:** Used for modifying the page DOM and injecting styles (e.g., blurring images) on content detection.
 - **Chrome Extensions API:** Interfaces with the Chrome browser to provide extension functionality.
 - [Chrome Extension API Documentation](#)

3.4 Project and Implementation Plan

1. Deliverables:

- ☐ Modules and Code:
 - User Interface (UI) Module: Fully developed UI that allows users to interact with the system for content moderation and control filtering preferences.
 - Queue Management Module: Code to manage the sequencing and processing of images/videos that need to be analyzed.
 - Detection Engine (NSFWX Model): Fine-tuned MobileNetV3-Small model integrated with TensorFlow.js for content detection.
 - Style Management Module: Code to apply visual effects (blurring, grayscale) based on detection results.
 - Settings Module: Code for managing user preferences and storing them in local storage or cloud storage.
 - Helper Functions: Utility code for image/video loading, resizing, and processing (helpers.js).

□ User Documentation:

- User Guide: Detailed instructions on how to install, configure, and use SafeView for content filtering and moderation.
- FAQs: A section to address common issues and troubleshooting.

2. Team Roles and Responsibilities and delivery schedule

Name of the Task	Developer	Tester	Approver	Date of Delivery
Module Development	Harikrishnan & Aditya	QA Team	Harikrishnan & Aditya	2025-04-01
User Interface Design	Harikrishnan	QA Team	Aditya	2025-03-01
Content Detection Engine (NSFWX)	Harikrishnan	QA Team	Aditya	2025-04-10
Settings and Preferences Management	Aditya	QA Team	Harikriishnan	2025-03-10
Style Management (Blur/Grayscale Effect)	Aditya	QA Team	Harikrishnan	2025-04-15
Testing and Quality Assurance	Harikrishnan & Aditya	QA Team	Harikrishnan & Aditya	2025-04-20
User Documentation Creation	Harikrishnan & Aditya	QA Team	Harikrishnan & Aditya	2025-08-23
Deployment and Installation Documentation	Harikrishnan	QA Team	Aditya	2025-04-23

Table 2: Team Roles

3. Risk Management Plan:

The SafeView project is subject to various risks that could impact its timeline, performance, and quality. Below are some identified risks and their mitigation strategies:

1. Model Performance Risks:
 - Risk: The NSFW detection model may not achieve the desired accuracy or might generate false positives/negatives.
 - Mitigation: Implement additional training and fine-tuning with an expanded dataset. Use techniques like data augmentation and regularization to improve

the model's performance. Implement a feedback loop from users to further refine the model.

2. Real-Time Processing Delays:
 - Risk: The system might experience latency during real-time content moderation due to large media files or complex processing.
 - Mitigation: Optimize the code for better performance by using techniques like batching, TensorFlow.js optimizations (e.g., quantization), and asynchronous processing for large media files.
3. User Experience Challenges:
 - Risk: Users may find the extension interface confusing or difficult to use, leading to low adoption.
 - Mitigation: Conduct usability testing to refine the interface and make it more user-friendly. Gather feedback during beta testing to improve the design.
4. Browser Compatibility Issues:
 - Risk: The extension may not be compatible with all browsers or versions, limiting its reach.
 - Mitigation: Perform thorough testing on multiple browsers (e.g., Chrome, Firefox, Edge) and ensure compatibility with the latest browser versions. Provide clear installation instructions for different environments.
5. Model Deployment and Integration Risks:
 - Risk: The NSFWX model might not function properly in the browser environment, leading to errors in content detection.
 - Mitigation: Test the model thoroughly in the TensorFlow.js environment before deployment. Ensure the model is optimized for in-browser use (e.g., quantization, pruning).
6. Security Risks:
 - Risk: The extension might collect or inadvertently share user data, violating privacy laws.
 - Mitigation: Ensure that all data is processed locally on the user's device and that no personal data is sent to external servers. Follow best practices for user privacy and comply with data protection regulations (e.g., GDPR).

3.5 Testing & Deployment Plan

1. Testing Strategy

A minimal testing strategy is outlined here. Detailed testing plans will be described in the dedicated Testing Section.

1. Unit Testing:
 - Each module (e.g., UI, Queue Management, Detection Engine, Style Management) will undergo unit testing to ensure the core logic functions correctly.
 - Tools: Jest for JavaScript unit testing.
2. Integration Testing:
 - Ensures that all modules interact correctly and that data flows seamlessly between them (e.g., from the UI to Queue Management, and then to the Detection Engine).
 - Tools: Mocha, Chai for integration testing.

3. System Testing:
 - Testing the entire SafeView system to ensure all components work together as expected in the browser environment.
 - Tools: Selenium or Cypress for end-to-end testing of the Chrome extension.
4. User Acceptance Testing (UAT):
 - Conducted with a sample user group to ensure the product meets end-user requirements and expectations.
 - Tools: Real users interacting with the system, tracking their feedback and performance issues.

2. Deployment Plan

1. Deployment Methods:

- The SafeView extension will be deployed through the Chrome Web Store.
- We will use continuous integration (CI) pipelines for regular updates and bug fixes.

2. Environment Setup:

- Development Environment:
 - Setup on local machines for development (Visual Studio Code, Node.js, TensorFlow.js).
 - Local testing of models before converting them for use in the browser.
- Production Environment:
 - Chrome Extension hosted on the Chrome Web Store.
 - TensorFlow.js models hosted locally or through a cloud storage service (e.g., Firebase).

3. Rollback Strategies:

- In case of a failed deployment, the previous stable version of the extension will be re-deployed from the backup.
- Backup: Ensure a stable version is available through a GitHub release, enabling rollback if the new deployment introduces critical bugs.

Chapter 4

Design Test Cases

This chapter details the testing conducted mentioning the various test scenarios considered. This chapter in a software development project report should provide a comprehensive overview of the testing process, methodologies, and results. The purpose of a test case document is to provide a detailed, structured record of the steps and conditions needed to verify a specific functionality or feature in software testing, ensuring that the application functions correctly and meets requirements

4.1 Introduction:

The SafeView project is focused on providing real-time, in-browser content moderation by detecting and filtering NSFW content on web pages. The success of this project hinges on thorough and effective testing, which ensures that the extension works seamlessly, and functions as expected across different scenarios.

These test cases target critical functionalities such as input validation, navigation within the extension, content detection accuracy, performance under edge cases, and real-time processing efficiency.

4.2 Test Cases

1. Validation of User Input Settings

Test Case ID	TC-001
Test Case Name	Validation of User Input Settings
Description	Validate that the extension accepts valid input for content filtering settings, including blur intensity and toggle switches.
Pre-Conditions	The SafeView Chrome extension is installed, and the user has accessed the settings page.
Test Steps	1. Open the SafeView extension settings page. 2. Enter valid values in the blur intensity slider (e.g., 0-100). 3. Toggle content filtering options 4. Save the changes.
Result	The settings are correctly saved, and the UI reflects the updates.
Pass/Fail Criteria	Pass if the settings are saved without errors and applied to content filtering.

Table 3: Test case TC-001

2. Navigation Between Settings and Content Filtering

Test Case ID	TC-002
Test Case Name	Navigation Between Settings and Content Filtering
Description	Ensure that users can access different sections of the settings and apply content filtering options.
Pre-Conditions	The SafeView extension is installed and active.
Test Steps	<ol style="list-style-type: none"> 1. Open the extension from the Chrome toolbar. 2. Navigate to the Settings section. 3. Navigate to the Content Filtering tab. 4. Toggle various filtering options. 5. Save the settings.
Expected Result	User can navigate through the settings, toggle filters, and save changes.
Pass/Fail Criteria	Pass if the user can navigate the UI and save preferences successfully.

Table 4: Test case TC-002

3. Verification of NSFW Content Detection

Test Case ID	TC-003
Test Case Name	Verification of NSFW Content Detection
Description	Verify that the model correctly classifies NSFW content and applies the correct filters.
Pre-Conditions	The SafeView extension is running, and content filtering is enabled.
Test Steps	<ol style="list-style-type: none"> 1. Load an image or video on a webpage. 2. Ensure the content is NSFW (e.g., explicit image or video). 3. The extension should detect the content and classify it as NSFW. 4. The Style Module should apply a blur effect to the detected content. 5. Verify that the content is blurred.
Expected Result	The content is detected as NSFW, and the appropriate visual filter (blur) is applied.
Pass/Fail Criteria	Pass if the NSFW content is detected, and the appropriate action (blurring) is applied.

Table 5: Test case TC-003

4. Verification of Safe Content Detection

Test Case ID	TC-004
Test Case Name	Verification of Safe Content Detection
Description	Ensure that the model correctly handles non-NSFW content and does not apply any filters.
Pre-Conditions	The SafeView extension is running, and content filtering is enabled in the settings.
Test Steps	<ol style="list-style-type: none"> 1. Load a non-NSFW image or video on a webpage. 2. The extension should classify the content as Safe. 3. Verify that no visual changes (e.g., blur or grayscale) are applied to the content. 4. The user receives a confirmation message (e.g., "Safe content detected").
Expected Result	The content is classified as Safe, and no filtering is applied.
Pass/Fail Criteria	Pass if the content is classified as safe and no visual changes occur.

Table 6: Test case TC-004

5. Edge Case Handling in Content Filtering

Test Case ID	TC-005
Test Case Name	Edge Case Handling in Content Filtering
Description	Verify the extension's ability to handle complex images or unusual sizes.
Pre-Conditions	The SafeView extension is installed and active.
Test Steps	<ol style="list-style-type: none"> 1. Load an image with complex content or an unusual size (e.g., very large, small, or heavily compressed image). 2. The extension should correctly analyze the content and classify it as either NSFW or safe. 3. Apply the appropriate action (e.g., blur effect for NSFW content, no action for safe content).
Expected Result	The extension should process the content correctly and apply appropriate filtering.
Pass/Fail Criteria	Pass if the extension correctly handles the edge case and applies the correct filtering.

Table 7: Test case TC-005

6. Real-Time Detection Performance

Test Case ID	TC-006
Test Case Name	Real-Time Detection Performance
Description	Ensure that content detection and filtering occurs quickly without noticeable delays in the user's browsing experience.
Pre-Conditions	The SafeView extension is active and configured for content filtering.
Test Steps	<ol style="list-style-type: none"> 1. Load a webpage with multiple media elements (images, videos). 2. Measure the time it takes to detect and apply the filter (e.g., blur) on all content. 3. Ensure there is no significant delay in page rendering or content display.
Expected Result	Content detection and filtering occur in real-time, with minimal delay.
Pass/Fail Criteria	Pass if the filtering occurs within an acceptable time frame (e.g., less than 2 seconds per content element).

Table 8: Test case TC-006

7. Handling of All Possible NSFW Content Types

Test Case ID	TC-007
Test Case Name	Handling of All Possible NSFW Content Types
Description	Ensure that the extension can handle every possible type of NSFW content, including emerging content formats.
Pre-Conditions	The SafeView extension is running with the model loaded.
Test Steps	<ol style="list-style-type: none"> 1. Load various types of media content with diverse NSFW content. 2. Check if all content types (e.g., explicit, deepfakes, evolving NSFW formats) are accurately detected. 3. Apply appropriate filters or actions based on classification.
Expected Result	All NSFW content types should be detected accurately and filtered appropriately.
Pass/Fail Criteria	Fail if any new or emerging NSFW content type is misclassified or missed by the system.

Table9: Test case TC-007

8. Real-Time Video Detection with Zero Latency

Test Case ID	TC-008
Test Case Name	Real-Time Video Detection with Zero Latency
Description	Ensure that the extension can process and filter video content without any detectable latency.
Pre-Conditions	The SafeView extension is running and content filtering is enabled.
Test Steps	<ol style="list-style-type: none"> 1. Load a webpage with a continuous video stream. 2. Enable content filtering and monitor the real-time detection. 3. Measure the time taken to process and apply filtering without affecting the video playback.
Expected Result	The video should be processed in real-time with no noticeable delay in playback.
Pass/Fail Criteria	Fail if there is any delay or lag in video processing.

Table 10: Test case TC-008

9. Seamless Cross-Browser Compatibility for All Web Platforms

Test Case ID	TC-009
Test Case Name	Seamless Cross-Browser Compatibility for All Web Platforms
Description	Ensure the SafeView extension works across all browsers (Chrome, Firefox, Safari, Edge) with the same functionality and UI behavior.
Pre-Conditions	The SafeView extension is installed and active on each browser.
Test Steps	<ol style="list-style-type: none"> 1. Install the SafeView extension on Chrome, Firefox, Safari, and Edge. 2. Enable content filtering and adjust settings on each browser. 3. Verify that the extension works consistently across all browsers.
Expected Result	The extension behaves identically across all browsers and applies filtering consistently.
Pass/Fail Criteria	Fail if any browser shows discrepancies in behavior, such as broken UI, missing features, or content filtering errors.

Table 11: Test case TC-009

4.3 Conclusion:

User Test Table for SafeView Project

Test Case ID	Test Case Name	Tester	Requirement Met
TC-001	Validation of User Input Settings	Harikrishnan	✓
TC-002	Navigation Between Settings and Content Filtering	Aditya	✓
TC-003	Verification of NSFW Content Detection	Harikrishnan	✓
TC-004	Verification of Safe Content Detection	Aditya	✓
TC-005	Edge Case Handling in Content Filtering	Harikrishnan	✓
TC-006	Real-Time Detection Performance	Aditya	✓
TC-007	Handling of All Possible NSFW Content Types	Harikrishnan	✗
TC-008	Real-Time Video Detection with Zero Latency	Aditya	✗
TC-009	Seamless Cross-Browser Compatibility for All Web Platforms	Harikrishnan	✗

Table 12: Test case completion

With the detailed test case design and clear assignment of responsibilities, the **SafeView** project is set to undergo rigorous testing to ensure that all features work as intended. The use of a structured approach to testing helps us maintain high software quality, minimize bugs, and optimize user experience. The results from this testing phase will inform any necessary improvements before deployment, ensuring that the extension delivers reliable and efficient content filtering for end-users.

Chapter 5

Implementation & Development of the Prototype

This chapter details the implementation process of the proposed prototype/application, outlining the development environment, tools, and technologies utilized. It describes the step-by-step approach taken to transform the conceptual design into a functional system. The implementation methodology, including coding standards, frameworks, and system architecture, is discussed in detail. Key challenges encountered during development and their corresponding solutions are also highlighted. Finally, the chapter concludes with an overview of the system's functionality and its readiness for testing and evaluation.

5.1 Introduction:

The System Implementation of the SafeView project relies on the architectural design and system analysis results to create a cohesive and functional system that meets the stakeholder and system requirements. The core elements of this system, including both the model training and the Chrome extension, will be developed and integrated to achieve the desired content moderation functionality for real-time NSFW detection. Below, we break down the implementation in two major sections: Model Training and Chrome Extension.

5.2 Modules Description:

Module	Name	Definition	Purpose
UI Module	Extension User Interface	Definition: Provides the user interface for the SafeView Chrome extension, allowing users to interact with the content filtering system.	Purpose: [Identifier: User Interaction, Name: UI Settings, Description: Displays user-configurable settings such as blur intensity, media types to filter (e.g., images, videos), and gender-based face detection, Type: Software application]
Queue Module	Queue Management	Definition: Manages the incoming media content (images, videos) to be processed for NSFW detection, ensuring that they are processed in sequence and efficiently.	Purpose: [Identifier: Data Management, Name: Media Queue, Description: Organizes and processes media content, sending them to the Content Detection Module for analysis, Type: Software application]

K. J. Somaiya School of Engineering
Department of Computer Engineering

Detection Module	Content Detection Engine	Definition: Analyzes the media content (images, videos) from the web pages using the trained NSFWX model to classify whether the content is safe or NSFW.	Purpose: [Identifier: Content Filtering, Name: Real-Time Detection, Description: Detects NSFW content in real-time and classifies it into categories like safe or NSFW, Type: Software application (TensorFlow.js)]
Style Module	Style Management	Definition: Applies CSS-based visual effects (such as blurring or grayscale) to the media content on the web page based on the classification results from the Content Detection Module.	Purpose: [Identifier: Visual Processing, Name: DOM Manipulation, Description: Modifies the DOM to apply blur or unblur effects to the detected media based on its classification, Type: Software application]
Event Module	Event Manager	Definition: Listens for events triggered by user interactions and propagates these events across different parts of the extension to update system behavior accordingly.	Purpose: [Identifier: Event Handling, Name: User Event Listener, Description: Listens for user interactions and propagates changes to Settings Management, Queue, Detection, and Style modules, Type: Software application]
Settings Module	Settings Manager	Definition: Stores and manages the user's preferences such as blur intensity, which types of content to filter, and site-specific settings (e.g., whitelisting certain websites).	Purpose: [Identifier: User Preferences, Name: Settings Storage, Description: Manages and applies user preferences to control content moderation settings, stores them persistently using chrome.storage.sync, Type: Software application]
Model Preprocessing Module	Image Preprocessing	Definition: Handles all preprocessing tasks for image data, including resizing, normalization, and augmentation.	Purpose: [Identifier: Data Preprocessing, Name: Image Resizing, Description: Resizes images to 224x224 and normalizes them for model input, Type: Software application]
Training Module	Neural Network Training	Definition: Defines the model architecture, compiles, and trains the MobileNetV3-Small model using the preprocessed LSPD dataset.	Purpose: [Identifier: Model Architecture, Name: MobileNetV3-Small Training, Description: Fine-tunes the MobileNetV3-Small model for classifying NSFW content, Type: Software application]
Model Evaluation Module	Model Evaluation	Definition: Evaluates the trained model's performance on the test set, generating metrics such as accuracy, precision, and recall.	Purpose: [Identifier: Performance Evaluation, Name: Model Evaluation, Description: Assesses the model's ability to generalize to unseen data, Type: Software application]

Model Conversion Module	TensorFlow.js Conversion	Definition: Converts the trained TensorFlow model to a format compatible with TensorFlow.js for in-browser inference.	Purpose: [Identifier: Model Deployment, Name: TensorFlow.js Conversion, Description: Prepares the trained model for use in the browser, Type: Software application]
--------------------------------	--------------------------	---	---

Table 13: Module Description

Model Training Modules

1. Model Preprocessing Module

- Definition: The Model Preprocessing Module handles all preprocessing tasks such as resizing images, normalizing pixel values, and augmenting the dataset. This is the first step before feeding the data into the neural network for training.
- Purpose:
 - Identifier: Image Preprocessing
 - Name: Data Augmentation and Preprocessing
 - Description: This module is responsible for preparing the images from the LSPD dataset for training by resizing them, normalizing pixel values, and applying data augmentation techniques.
 - Type: Software application
- Activity:
 - Input: Image data from the LSPD dataset (raw images in their original resolution).
 - Output: Processed image data with resized dimensions (224x224), normalized pixel values, and augmented images.
 - Functionality:
 - Resizes images to fit the input dimensions of the MobileNetV3-Small model (224x224 pixels).
 - Normalizes image pixel values to be between 0 and 1.
 - Applies random augmentations like flips, rotations, and adjustments in brightness to enhance the model's ability to generalize.

2. Training Module

- Definition: The Training Module defines the neural network architecture, compiles the model, and trains the model on the preprocessed dataset. It is responsible for building the model and optimizing it based on the training data.
- Purpose:
 - Identifier: Model Architecture and Training
 - Name: Neural Network Training
 - Description: This module sets up the MobileNetV3-Small model, compiles it with an appropriate optimizer, and trains it using the preprocessed image data.
 - Type: Software application
- Activity:
 - Input: Preprocessed image data, one-hot encoded labels, training parameters (e.g., learning rate, batch size).

- Output: Trained model weights, performance metrics (accuracy, precision, recall, F1-score).
- Functionality:
 - Defines the MobileNetV3-Small architecture for feature extraction.
 - Adds custom dense layers to classify images into 5 categories.
 - Compiles the model with the Adam optimizer and categorical cross-entropy loss function.
 - Trains the model for multiple epochs, using the validation set to monitor performance and adjust hyperparameters.

3. Model Evaluation Module

- Definition: The Model Evaluation Module evaluates the performance of the trained model using the test dataset. It measures the effectiveness of the model through various metrics such as accuracy, precision, recall, and F1-score.
- Purpose:
 - Identifier: Performance Evaluation
 - Name: Model Evaluation
 - Description: This module evaluates the model's classification performance on unseen data to ensure it generalizes well.
 - Type: Software application
- Activity:
 - Input: Test dataset images, true labels.
 - Output: Performance metrics (accuracy, confusion matrix, precision, recall).
 - Functionality:
 - Computes accuracy by comparing model predictions against true labels.
 - Generates a confusion matrix to visualize class-wise performance.
 - Outputs precision, recall, and F1-score for a detailed evaluation of model performance.

4. Model Conversion Module

- Definition: The Model Conversion Module converts the trained TensorFlow model to a TensorFlow.js-compatible format to be used in the SafeView Chrome extension for real-time content detection.
- Purpose:
 - Identifier: TensorFlow.js Model Conversion
 - Name: Model Conversion for In-Browser Inference
 - Description: This module handles the conversion of the trained TensorFlow model into a TensorFlow.js-compatible format.
 - Type: Software application
- Activity:
 - Input: Trained TensorFlow model.
 - Output: TensorFlow.js model files (model.json, weight files).
 - Functionality:
 - Converts the trained model using TensorFlow.js conversion tools.

- Optimizes the model for in-browser use, including pruning and quantization for faster inference.

Chrome Extension Modules

1. UI Module (Extension User Interface)

- **Definition:** The UI Module is the graphical interface that allows users to interact with the SafeView Chrome extension. This module displays the settings and preferences that users can modify, such as toggling content filtering, adjusting blur intensity, and enabling gender-based face detection.
- **Purpose:**
The primary purpose of the UI Module is to facilitate user interaction with the extension. It presents an intuitive interface where users can configure settings related to content moderation (blur intensity, grayscale, etc.). Additionally, it provides real-time feedback on the status of the content being filtered.
- **Activity:**
 - **Input:** The UI collects user preferences, such as toggles for enabling/disabling content filtering and sliders for adjusting blur intensity.
 - **Output:** It updates the user preferences in `chrome.storage.sync`, allowing the settings to be persistent across sessions. The UI also displays real-time updates based on these preferences.

2. Queue Management Module

- **Definition:** The Queue Management Module handles the queuing and sequencing of media elements (images and videos) that need to be processed. This module ensures that the content is processed in an orderly fashion, preventing system overload and ensuring each media element is classified before moving to the next.
- **Purpose:** The Queue Management Module plays a critical role in ensuring the extension runs efficiently. It manages the flow of media content and provides a controlled environment for content detection to be processed sequentially.
- **Activity:**
 - **Input:** The Queue Management Module receives media content (images/videos) from the UI Module or the DOM of web pages.
 - **Output:** The content is then sent to the Detection Module for analysis. After processing, results are passed to the Style Module for visual updates (e.g., applying blur or grayscale).

3. Content Detection Module

- **Definition:** The Content Detection Module is responsible for detecting and classifying media content (images and videos) on web pages. It uses the trained NSFWX model, based on MobileNetV3-Small, to classify whether the content is NSFW or safe.
- **Purpose:** The Content Detection Module processes incoming images and videos in real-time. It classifies each piece of content as NSFW or safe and determines what action should be taken (e.g., blur, grayscale, etc.).

- Activity:
 - Input: The Detection Module receives media content (images/videos) from the Queue Management Module.
 - Output: The module outputs classification results (e.g., NSFW, Safe) and metadata (e.g., probability scores for each classification) that are sent to the Style Management Module for visual adjustments.

4. Style Management Module

- Definition:

The Style Management Module is responsible for applying CSS-based visual effects (such as blurring or grayscale) to the media content based on the classification results from the Content Detection Module. It modifies the DOM to visually filter content in real-time.
- Purpose:

The Style Module ensures that the user's preferences are visually applied to the content. Based on the results from the Detection Module, this module either blurs NSFW content or applies other visual changes (like grayscale).
- Activity:
 - Input: The Style Module receives classification results from the Detection Module (e.g., NSFW, safe).
 - Output: It applies CSS styles (blur, grayscale) to the corresponding media elements on the web page.

5. Event Module

- Definition:

The Event Module listens for events triggered by user interactions and propagates these events across different parts of the extension to update system behavior. It handles updates such as enabling/disabling content filtering, adjusting blur intensity, and toggling other features.
- Purpose:

The Event Module is responsible for ensuring that user interactions with the UI Module trigger the appropriate changes in the extension's functionality. It manages the flow of events to ensure the content filtering system works according to the user's settings.
- Activity:
 - Input: User interactions from the UI Module (e.g., toggles, sliders).
 - Output: Propagates these events to the appropriate modules, such as Settings Module (to store preferences) and Detection Module (to update the content filtering process).

6. Settings Module

- **Definition:**
The Settings Module is responsible for managing and storing user preferences related to content moderation (e.g., blur intensity, which content types to filter, and site-specific settings). This module ensures that settings are persistent and available across sessions.
- **Purpose:**
The Settings Module stores and manages user-defined settings. It ensures that the preferences are applied consistently across sessions and modules, such as toggling content filtering and adjusting blur intensity.
- **Activity:**
 - **Input:** User settings such as blur intensity, content filter toggles, gender-based filtering.
 - **Output:** Saves user preferences in `chrome.storage.sync` for persistence. It provides these preferences to other modules (e.g., Queue Module, Detection Module) for consistent behavior.

5.3 Integration:

Integration is a critical step in the implementation of the SafeView project, ensuring that all components (modules) function cohesively to meet the system's requirements. During the integration phase, various modules, such as the UI, Queue, Detection, Style, Event, Settings, and Model Training, are combined to form a complete and functional system. The goal is to ensure that the project produces the expected output, with all modules interacting correctly and efficiently.

Integration Strategy:

1. Module Communication:

- a. UI Module communicates with the Settings Module to retrieve and store user preferences (e.g., blur intensity, content filtering preferences).
- b. The Queue Module receives content (images/videos) from the UI Module and sends it for processing by the Detection Module.
- c. The Detection Module uses the NSFWX model (trained in the Model Training Module) to classify content as safe or NSFW. The classification results are passed to the Style Module.
- d. The Style Module applies CSS styles (such as blurring or grayscale) to the content on the web page, depending on the classification results from the Detection Module.
- e. The Event Module listens for user interactions with the UI Module, propagating those changes to the relevant modules to update behavior dynamically.

2. Data Flow:

- a. **User Inputs:** The user interacts with the UI, adjusting settings (e.g., toggling content filtering, adjusting blur intensity).
- b. **Settings Storage:** The Settings Module stores and synchronizes the user preferences, making them accessible to other modules.
- c. **Content Processing:** The Queue Module receives media elements, which are then passed on to the Detection Module. The Detection Module processes the content using the NSFWX model, classifies it, and sends the results to the Style Module for visual modifications.
- d. **Output:** The Style Module applies visual changes (e.g., blurring, grayscale) based on classification results, modifying the web page content.

3. Dependency Management:

- a. **TensorFlow.js:** The Detection Module depends on TensorFlow.js for in-browser model inference. This allows the trained NSFWX model (converted to TensorFlow.js format) to be used directly in the browser for real-time content moderation.
- b. **chrome.storage.sync:** The Settings Module depends on chrome.storage.sync to store user preferences persistently, ensuring that settings are applied across sessions.
- c. **TensorFlow.js Model Files:** The Model Conversion Module converts the trained TensorFlow model to TensorFlow.js compatible files (model.json and shard files). These are used by the Detection Module to run inference directly in the browser.

4. Version Control:

- a. Each module of the SafeView project is version-controlled using Git. This ensures that changes to the code are tracked, and previous versions can be accessed if needed.
- b. Any updates to the NSFWX model or TensorFlow.js dependencies will be documented and versioned, ensuring that compatibility is maintained across the extension and the trained model.

5. Integration Testing:

- a. During integration, each module is tested for its functionality, ensuring that inputs flow correctly between modules and the expected outputs are generated.
- b. Unit testing is done for individual modules (e.g., UI Module, Queue Module) to ensure they function independently.
- c. End-to-end testing ensures that all modules work together, and real-time content filtering is performed accurately.
- d. Regression testing ensures that changes or updates do not break existing functionality.

5.4 Dataset

The LSPD (Large-Scale Pornographic Dataset for Detection and Classification) is a unique and comprehensive dataset designed for the detection and classification of pornographic content. It includes a wide variety of high-quality labeled images and videos annotated with sexual objects, enabling object detection and classification tasks for deep learning models. Below is the detailed process of how the dataset was selected and acquired, along with sample data and a data dictionary.

Dataset Source:

- Source: The dataset was introduced in the paper titled "LSPD: A Large-Scale Pornographic Dataset for Detection and Classification," authored by Dinh Duy Phan and colleagues, published in the International Journal of Intelligent Engineering and Systems, Vol.15, No.1, 2022. The paper presents the LSPD dataset as the largest public dataset available for both pornographic image/video classification and sexual object detection/segmentation tasks.
- Link to Dataset: The dataset is publicly available for research purposes, and can be accessed through the provided link in the paper: [LSPD Dataset Access](#)

Dataset Overview and Structure:

- Number of Images: 500,000 images (divided into categories such as Porn, Non-porn, Sexy, Hentai, and Drawing).
- Number of Videos: 4,000 videos (divided into Porn and Non-porn categories).
- Annotation Details:
 - The dataset provides polygon masks for four sexual objects: Male Genitals, Female Genitals, Breast, and Anus.
 - Images are labeled with five categories for classification: Porn, Non-porn, Sexy, Hentai, and Drawing.

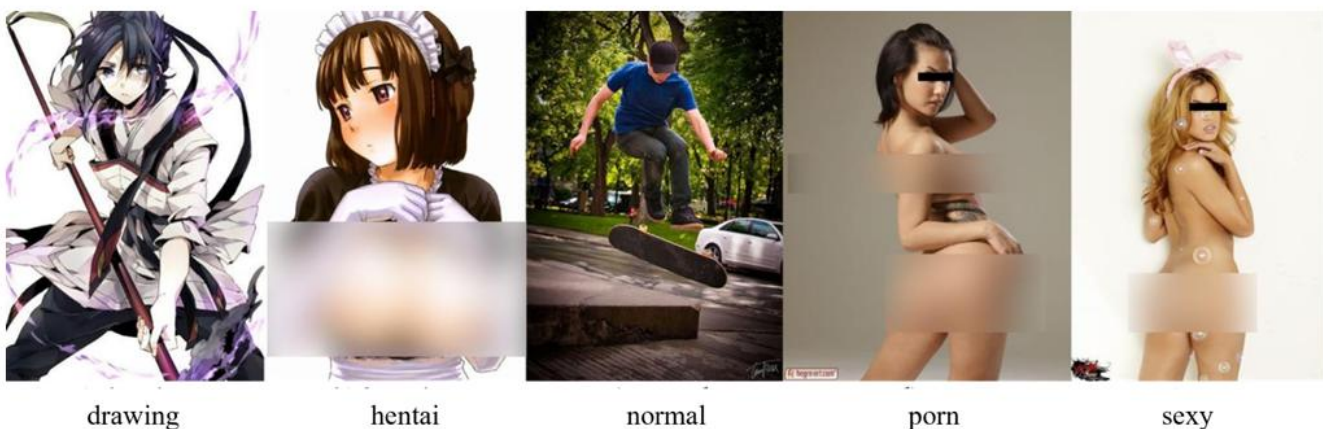


Figure. 1 Sample images in our dataset (left to right): drawing, hentai, non-porn, porn, and sexy

Figure 5: LSPD Dataset sample

Process of Dataset Selection and Acquisition:

1. Initial Research and Selection:
 - The need for a high-quality dataset that includes both image classification and object detection tasks for pornography detection was identified.
 - Existing Datasets: Several pornographic datasets were available but lacked high resolution and recent content. Some datasets were limited in scope, either focusing only on classification or lacking detailed object annotations.
2. Contacting the Authors:
 - After reviewing the paper and understanding the dataset's scope, we reached out to the authors for access to the LSPD dataset.
 - An email was sent to the corresponding author (Duc-Lung Vu) to request permission to use the dataset for research purposes, specifying how the dataset would be used in the SafeView project for NSFW content detection.
 - The dataset was shared with us following approval, and all necessary terms and conditions for its usage were clarified.
3. Data Upload:
 - After acquiring the dataset, we uploaded it to the HPC system for processing and usage in model training.
 - The dataset was split into three subsets for training, validation, and testing as follows:
 - Training Set: 350,000 images (70% of total)
 - Validation Set: 75,000 images (15% of total)
 - Test Set: 75,000 images (15% of total)
 - These subsets were uploaded onto the HPC system for further use in model training.

5.5 Implementation details

The SafeView project involves two primary components: model training and the Chrome extension. Below are the implementation details for both.

Model Training Implementation

1. Dataset Preparation:
 - 50 Percent of The LSPD dataset (250,000 images) is used for training.
 - Preprocessing: Images are resized to 224x224 pixels and normalized. Data augmentation is applied to improve generalization.
2. Model Architecture:
 1. Base Model: MobileNetV3-Small (pre-trained on ImageNet) is used for feature extraction.
 2. Custom Layers: Described in the table

Layer	Purpose	Importance	Explanation / Role in the Model
Global Average Pooling2D	Reduces the spatial dimensions of the feature map from the backbone.	Essential for converting feature maps into a fixed-size vector.	This layer takes the average of each feature map across all spatial dimensions (height and width) to produce a 1D tensor (vector) representing the learned features for each image.
Dense(256, ReLU)	Fully connected layer with 256 units and ReLU activation.	Critical for introducing non-linearity and increasing model capacity.	This layer is responsible for learning complex patterns and interactions between the extracted features from the previous pooling operation. The ReLU activation introduces non-linearity.
Dropout(0.5)	Regularization layer that randomly drops 50% of neurons during training.	Helps prevent overfitting and improves generalization.	This layer randomly drops 50% of the neurons during training to prevent overfitting, forcing the model to rely on multiple different combinations of neurons to make predictions.
Dense(128, ReLU)	Fully connected layer with 128 units and ReLU activation.	Further reduces the dimensionality of the model's features.	Similar to the previous Dense layer but with fewer units, this layer reduces the complexity of the learned representations and further refines the extracted features for classification.
Dropout(0.3)	Regularization layer that randomly drops 30% of neurons during training.	Reduces the risk of overfitting in a deeper network.	This layer randomly drops 30% of the neurons in the previous Dense layer during training, ensuring that the model does not become too reliant on any single feature for making predictions.
Dense(NUM CLASSES, Softmax)	Output layer with softmax activation for classification.	Converts the model output into probabilities for each class.	The final output layer provides probabilities for each of the <code>NUM_CLASSES</code> . The Softmax activation ensures the outputs sum to 1, allowing the model to interpret the result as a probability distribution over the classes.

Table 14: Custom Layers in CNN

In this study, we tested five distinct approaches to train and fine-tune a MobileNetV3-Small model for NSFW content classification. Each approach was designed to evaluate the impact of different fine-tuning strategies and backbone adaptation methods on the model's performance. Below are the details of the five approaches that were tested:

1. Feature-Extractor Baseline

The model only trains the custom head layers while keeping the entire MobileNetV3 backbone frozen, providing a baseline performance.

Aspect	No Fine-Tuning (Training)	Fine-Tuning Strategy	Inferences
Custom Layers (Head)	<ul style="list-style-type: none"> - GlobalAveragePooling2D - Dense(256, ReLU) - Dropout(0.5) - Dense(128, ReLU) - Dropout(0.3) - Dense(NUM_CLASSES, Softmax) 	Used in fine-tuning — Always trainable	The custom head layers are consistent and always updated. They focus on NSFW-specific classification.
Backbone (MobileNet V3)	Frozen – All layers in the MobileNetV3 backbone are kept frozen during training. Only the custom head is trained.	No fine-tuning – Backbone layers remain frozen.	Baseline for comparison: Freezing the backbone lets the model quickly adapt to task-specific features without altering pre-trained feature maps.
Training Strategy	25 epochs, batch_size = 32, Adam(lr = 1e-3)	N/A – No fine-tuning applied.	The focus is solely on training the custom head with the pre-trained backbone, acting as the "control" model for subsequent fine-tuning experiments.
Class-Weight Strategy	Inverse-frequency class weights computed from dataset.	No change — class weights remain unchanged.	Handling class imbalance ensures that the model doesn't become biased toward the majority class.
Callbacks	Early stopping with patience 5, model checkpoint on validation accuracy.	N/A – No fine-tuning.	Standard callbacks to prevent overfitting.
Risks & Limitations	Low risk of overfitting due to frozen backbone, but potentially underperforms in complex tasks.	N/A	Baseline is a good starting point but likely to have lower accuracy compared to fine-tuned models.

Table 15: CNN model training – Approach 1

2. Late-Block Fine-Tuning

The model unfreezes and fine-tunes the last 30 layers of the MobileNetV3 backbone while keeping the earlier layers frozen.

Aspect	No Fine-Tuning (Training)	Fine-Tuning Strategy	Inferences
Custom Layers (Head)	<ul style="list-style-type: none"> - GlobalAveragePooling2D - Dense(256, ReLU) - Dropout(0.5) - Dense(128, ReLU) - Dropout(0.3) - Dense(NUM_CLASSES, Softmax) 	Used in fine-tuning — Always trainable	The custom head layers are consistent and always updated.
Backbone (MobileNetV3)	Frozen – All MobileNetV3 layers are frozen, only the custom head is trained.	Last 30 layers unfrozen – Fine-tuning only the last 30 layers of MobileNetV3.	Selective unfreezing allows for faster adaptation of deeper features while retaining the generalization from frozen early layers.
Training Strategy	25 epochs, batch_size = 128, Adam(lr = 1e-3)	25 additional epochs, Adam(lr = 5e-4), only fine-tuning last 30 layers.	Fine-tuning the deeper layers allows the model to adapt to task-specific patterns without altering the core general features of MobileNetV3.
Class-Weight Strategy	Inverse-frequency class weights computed from dataset.	No change — class weights remain unchanged.	Fine-tuning uses the same class weights, ensuring consistency in handling class imbalance.
Callbacks	Early stopping with patience 5, model checkpoint on validation accuracy.	Early stopping with patience 5, model checkpoint on validation accuracy.	The same callbacks are used for both training and fine-tuning.
Risks & Limitations	Moderate risk of overfitting, as the backbone is frozen, but it's trained solely for specific features.	Fine-tuning only the last 30 layers introduces moderate risk of overfitting, but prevents large-scale forgetting.	Fine-tuning the last few layers focuses on task-specific adjustments, providing a good balance between generalization and specialization.

Table 16: CNN model training – Approach 2

3. Selective Fine-Tuning with Layer Blocks

This approach selectively unfreezes specific blocks of the MobileNetV3 backbone, focusing on mid and high-level features for fine-tuning.

Aspect	No Fine-Tuning (Training)	Fine-Tuning Strategy	Inferences
Custom Layers (Head)	<ul style="list-style-type: none"> - GlobalAveragePooling2D - Dense(256, ReLU) - Dropout(0.5) - Dense(128, ReLU) - Dropout(0.3) - Dense(NUM_CLASSES, Softmax) 	Used in fine-tuning — Always trainable	Custom layers are always trained in both phases, ensuring task-specific learning.
Backbone (MobileNetV3)	Frozen – All MobileNetV3 layers are frozen during training.	Selective unfreezing – Unfreezes blocks 5, 7, and all layers ≥ 8 .	Selective fine-tuning enables more targeted adaptation to mid and high-level features while preserving lower-level feature extraction.
Training Strategy	25 epochs, batch_size = 128, Adam(lr = 1e-3)	25 epochs, Adam(lr = 3e-4), fine-tuning selected blocks of the backbone.	By focusing on mid-level and high-level blocks, the model adjusts to more complex patterns while retaining lower-level feature extraction.
Class-Weight Strategy	Inverse-frequency class weights computed from dataset.	No change — class weights remain unchanged.	The same class weights ensure balanced treatment of all classes during training and fine-tuning.
Callbacks	Early stopping with patience 5, model checkpoint on validation accuracy.	Early stopping with patience 5, model checkpoint on validation accuracy.	Ensures that the fine-tuning phase is effectively controlled to prevent overfitting.
Risks & Limitations	Moderate risk of overfitting, as only the head is trained.	Fine-tuning only specific blocks balances adaptation with retention of early features.	Key advantage: Allows targeted learning without losing general features, disadvantage: might not fully adapt to complex patterns.

Table 17: CNN model training – Approach 3

4. Full Backbone Fine-Tuning with Discriminative Learning Rates

All layers of the MobileNetV3 backbone are unfrozen, and discriminative learning rates are applied to different layers to control the adaptation speed.

Aspect	No Fine-Tuning (Training)	Fine-Tuning Strategy	Inferences
Custom Layers (Head)	<ul style="list-style-type: none"> - GlobalAveragePooling2D - Dense(256, ReLU) - Dropout(0.5) - Dense(128, ReLU) - Dropout(0.3) - Dense(NUM_CLASSES, Softmax) 	Used in fine-tuning — Always trainable	Custom layers always remain trainable, and their weights are updated during both phases.
Backbone (MobileNetV3)	Frozen – All layers in MobileNetV3 are frozen during training.	All layers unfrozen – Fine-tuning all layers of MobileNetV3.	Fine-tuning all layers allows for maximum flexibility in feature adaptation but risks overfitting without careful LR control.
Training Strategy	25 epochs, batch_size = 128, Adam(lr = 1e-3)	30 epochs, Adam(lr = 5e-5) for all layers with discriminative learning rates (lower LR for earlier layers).	Discriminative LR (e.g., early layers train slower) ensure effective adaptation of deeper features while retaining earlier learned features.
Class-Weight Strategy	Inverse-frequency class weights computed from dataset.	No change — class weights remain unchanged.	Class weights are consistent, ensuring that no class is overlooked during both phases.
Callbacks	Early stopping with patience 5, model checkpoint on validation accuracy.	Early stopping with patience 8, model checkpointing on best validation accuracy.	Additional patience in fine-tuning ensures that more epochs can be run without overfitting.
Risks & Limitations	Moderate risk of overfitting, as only the head is trained.	Fine-tuning all layers can lead to overfitting if not carefully managed. Discriminative LR help mitigate this risk.	Key benefit: All features are adapted, but it requires fine control over the learning rate to prevent catastrophic forgetting and overfitting.

Table 18: CNN model training – Approach 4

5. Progressive Fine-Tuning with Phases

The model fine-tunes the MobileNetV3 backbone in phases, progressively unfreezing layers as training progresses to ensure gradual feature adaptation.

Aspect	No Fine-Tuning (Training)	Fine-Tuning Strategy	Inferences
Custom Layers (Head)	<ul style="list-style-type: none"> - GlobalAveragePooling2D - Dense(256, ReLU) - Dropout(0.5) - Dense(128, ReLU) - Dropout(0.3) - Dense(NUM_CLASSES, Softmax) 	Used in fine-tuning — Always trainable	Custom layers remain consistent and always updated.
Backbone (MobileNetV3)	Frozen – All MobileNetV3 layers frozen during training.	Progressive unfreezing – Starts with no layers unfrozen, and progressively unfreezes the last blocks in 4 phases.	A gradual unfreezing process reduces the risk of catastrophic forgetting and ensures that layers are adapted when needed.
Training Strategy	25 epochs, batch_size = 1024, Adam(lr = 3e-4)	4 phases of training with LR decay per phase: Epochs = [5, 5, 8, 10] per phase.	Phased training helps adapt progressively, allowing the model to start with general features and slowly specialize in complex patterns.
Class-Weight Strategy	Inverse-frequency class weights computed from dataset.	Smoothed class weights (to avoid extreme values).	Smoothed weights reduce the overemphasis on rare classes that can distort model learning, improving robustness.
Callbacks	Early stopping with patience 5, model checkpoint on validation accuracy.	Early stopping with patience 5, model checkpointing on best validation accuracy.	This approach monitors each phase independently, ensuring effective control over fine-tuning and early stopping if validation accuracy stagnates.
Risks & Limitations	Moderate risk of overfitting, as only the head is trained.	Progressive fine-tuning mitigates catastrophic forgetting but requires careful control of learning rates.	Key advantage: Gradual learning adapts to the task without overwhelming the model. However, longer training times are required.

Table 19: CNN model training – Approach 5

4. Model Evaluation:

- Evaluated on a test set (4500 images) using accuracy, precision, recall, and confusion matrix.

5. Model Conversion:

- The trained model is converted to TensorFlow.js format for use in the extension.

Chrome Extension Implementation

1. User Interface (UI):

- UI Elements: Includes toggle switches, sliders, and checkboxes for controlling filtering preferences (blur intensity, content types).
- Interaction: The UI Module communicates with the Settings Module to save and retrieve preferences.

2. Content Detection:

- Model: The NSFWX model is loaded using TensorFlow.js for real-time detection of NSFW content.
- Process: Media (images/videos) from the web page is classified as safe or NSFW.

3. Style Management:

- Blur Effect: If content is classified as NSFW, the Style Module applies a blur effect to the media using CSS.

4. Event Handling:

- Event Module listens for user changes (e.g., toggling filters) and propagates updates to the relevant modules.

5. Settings Management:

- Storage: User settings are saved in chrome.storage.sync for persistence.
- Updates: Changes in settings are broadcast to other modules (e.g., Queue, Detection, Style).

Testing and Deployment

1. Testing:

- Unit Testing: Ensures each module functions independently.
- Integration Testing: Verifies that all modules work together seamlessly.
- User Acceptance Testing (UAT): Ensures the extension meets user needs and is intuitive to use.

3. Rollback Strategy:

- In case of issues, a previous stable version can be restored, leveraging Git version control for tracking changes.

5.6 Implementation Screenshots

Extension UI:

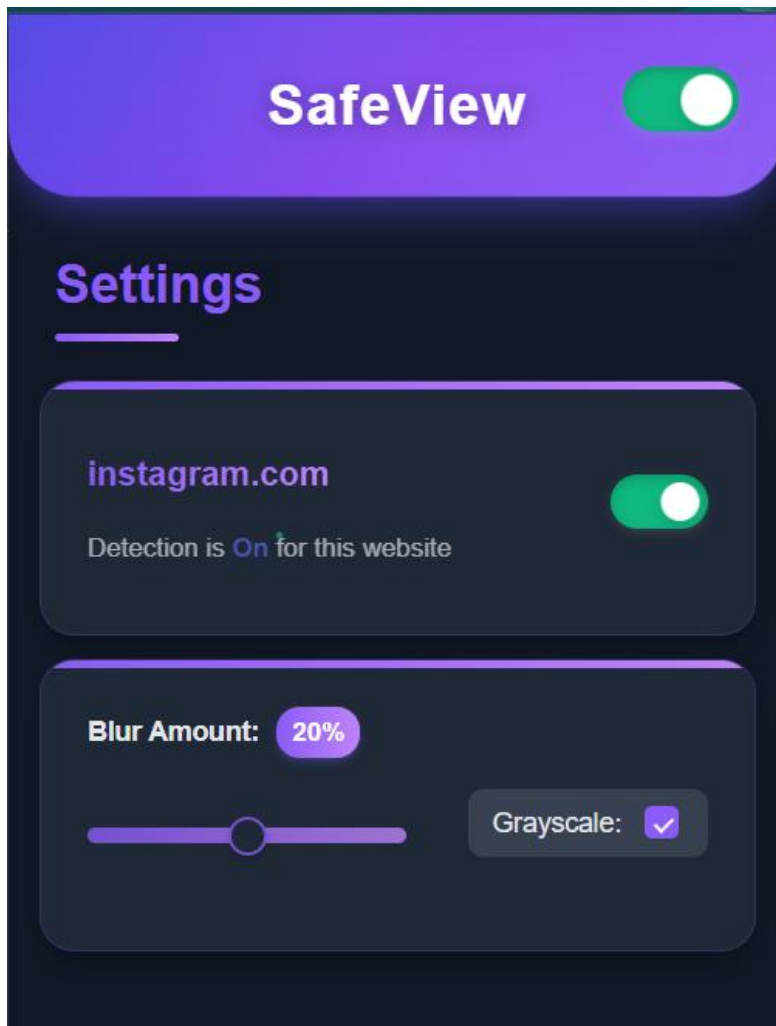
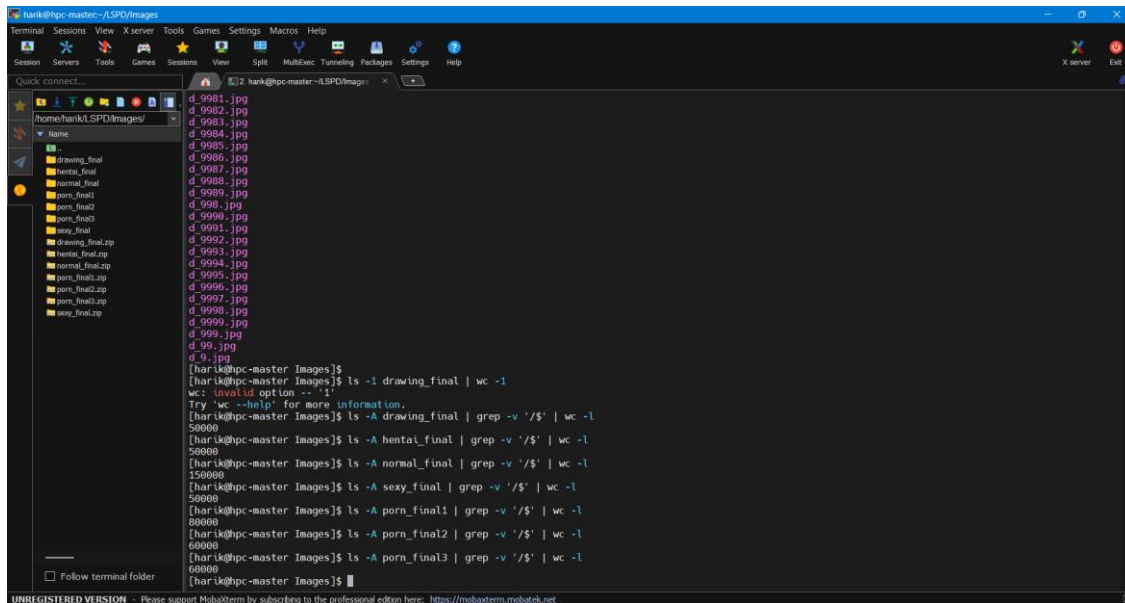


Figure 6: Extension UI

HPC environment:

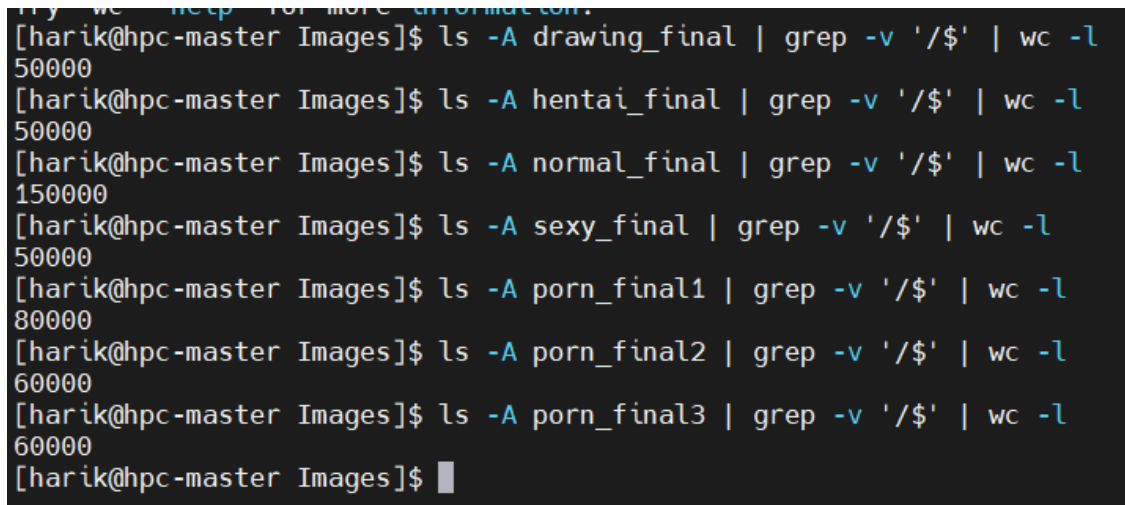


```

harik@hpc-master:~/LSPD/Images
ls
d_9981.jpg
d_9982.jpg
d_9983.jpg
d_9984.jpg
d_9985.jpg
d_9986.jpg
d_9987.jpg
d_9988.jpg
d_9989.jpg
d_9990.jpg
d_9991.jpg
d_9992.jpg
d_9993.jpg
d_9994.jpg
d_9995.jpg
d_9996.jpg
d_9997.jpg
d_9998.jpg
d_9999.jpg
d_99.jpg
d_9.jpg
[harik@hpc-master Images]$
[harik@hpc-master Images]$ ls -l drawing_final | wc -l
wc: invalid option -- 'l'
Try 'wc --help' for more information.
[harik@hpc-master Images]$ ls -A drawing_final | grep -v '/$' | wc -l
50000
[harik@hpc-master Images]$ ls -A hentai_final | grep -v '/$' | wc -l
50000
[harik@hpc-master Images]$ ls -A normal_final | grep -v '/$' | wc -l
150000
[harik@hpc-master Images]$ ls -A sexy_final | grep -v '/$' | wc -l
50000
[harik@hpc-master Images]$ ls -A porn_final1 | grep -v '/$' | wc -l
80000
[harik@hpc-master Images]$ ls -A porn_final2 | grep -v '/$' | wc -l
60000
[harik@hpc-master Images]$ ls -A porn_final3 | grep -v '/$' | wc -l
60000
[harik@hpc-master Images]$
  
```

Figure 7: HPC Environment

Dataset loading:



```

[harik@hpc-master Images]$ ls -A drawing_final | grep -v '/$' | wc -l
50000
[harik@hpc-master Images]$ ls -A hentai_final | grep -v '/$' | wc -l
50000
[harik@hpc-master Images]$ ls -A normal_final | grep -v '/$' | wc -l
150000
[harik@hpc-master Images]$ ls -A sexy_final | grep -v '/$' | wc -l
50000
[harik@hpc-master Images]$ ls -A porn_final1 | grep -v '/$' | wc -l
80000
[harik@hpc-master Images]$ ls -A porn_final2 | grep -v '/$' | wc -l
60000
[harik@hpc-master Images]$ ls -A porn_final3 | grep -v '/$' | wc -l
60000
[harik@hpc-master Images]$
  
```

Figure 8: Dataset Loading

MobileNet model architecture loading:

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 224, 224, 3)	0	-
rescaling (Rescaling)	(None, 224, 224, 3)	0	input_layer[0][0]
conv (Conv2D)	(None, 112, 112, 16)	432	rescaling[0][0]
conv_bn (BatchNormalization)	(None, 112, 112, 16)	64	conv[0][0]
activation (Activation)	(None, 112, 112, 16)	0	conv_bn[0][0]
expanded_conv_depthwise_pad (ZeroPadding2D)	(None, 113, 113, 16)	0	activation[0][0]
expanded_conv_depthwise (DepthwiseConv2D)	(None, 56, 56, 16)	144	expanded_conv_depthwise_p...
expanded_conv_depthwise_bn (BatchNormalization)	(None, 56, 56, 16)	64	expanded_conv_depthwise[0...
re_lu (ReLU)	(None, 56, 56, 16)	0	expanded_conv_depthwise_b...
expanded_conv_squeeze_excite... (GlobalAveragePooling2D)	(None, 1, 1, 16)	0	re_lu[0][0]
expanded_conv_squeeze_excite... (Conv2D)	(None, 1, 1, 8)	136	expanded_conv_squeeze_exc...

Figure 9: MobileNet V3 Model Architecture

Model Training Progress with Accuracy and Loss Metrics in the First Epoch:

call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `us`)`, as they will be ignored. self.warn_if_super_not_called() Epoch 1/30 5/612 ————— 14:47 1s/step - accuracy: 0.2001 - loss: 1.6755	Ln 38, Col 1
call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `us`)`, as they will be ignored. self.warn_if_super_not_called() Epoch 1/30 5/612 ————— 14:47 1s/step - accuracy: 0.2001 - loss: 1.6755	Ln 38, Col 1

Figure 10: Running Epochs

Saved Model in h5 fafter training process is complete:

Name	Date modified	Type	Size
Yesterday			
convert.py	17-04-2025 23:05	Python Source File	2 KB
nsfw_mobilenetv3_model.h5	17-04-2025 14:02	H5 File	10,909 KB
nsfw_mobilenetv3_model.json	17-04-2025 14:02	JSON Source File	150 KB

Figure 11: Saved Model in h5 format

Tensorflowjs format model after conversion:

Name	Date modified	Type	Size
group1-shard1of2.bin	17-04-2025 23:13	BIN File	4,096 KB
group1-shard2of2.bin	17-04-2025 23:13	BIN File	281 KB
model.json	17-04-2025 23:13	JSON Source File	153 KB
manifest.json	22-03-2025 15:58	JSON Source File	2 KB
package.json	22-03-2025 15:26	JSON Source File	1 KB
package-lock.json	22-03-2025 15:04	JSON Source File	166 KB
.gitignore	20-03-2025 11:23	Git Ignore Source ...	1 KB
vite.config.js	20-03-2025 11:23	JavaScript Source ...	1 KB
node_modules	18-04-2025 20:05	File folder	
src	18-04-2025 20:05	File folder	
tfjs	18-04-2025 20:05	File folder	
dist	18-04-2025 20:05	File folder	

Figure 12: Conversion of model to tensorflowjs

Rest of Implementation details are in the Results section. (including final accuracy and other measures after completion of model training.)

5.7 Git link

GitHub Repository Link: <http://github.com/hk151109/SafeView>

Description:

The **SafeView** GitHub repository contains the source code for the **SafeView Chrome Extension** and the underlying **NSFWX model**. It includes code for model training, extension development, and real-time content filtering. The repository serves as the central hub for the development, version control, and future updates of the project. It also provides the required setup instructions, dependencies, and detailed explanations for replicating and extending the project.

Chapter 6

Result and discussion

This chapter presents the results obtained from the implementation of the prototype/application and provides an in-depth analysis of its performance. The findings are evaluated based on predefined metrics, including accuracy, efficiency, usability, and reliability. A comparative analysis with existing systems or methodologies is conducted to assess improvements and innovations introduced by the proposed approach. Additionally, key observations, challenges encountered during testing, and their implications are discussed.

1.1 Results

Model development Results

In this experiment, five different approaches were implemented and tested for training and fine-tuning a MobileNetV3-Small model for NSFW content classification. Each approach explores a different strategy for handling the backbone model, with variations in fine-tuning, layer freezing, and learning rates. The following five approaches were tested, and their performance metrics are summarized and compared in the sections below.

Approach	Accuracy	Class	Precision	F1	Recall
Feature-Extractor Baseline (No Fine-Tuning)	0.8614	Drawing	0.72	0.80	0.91
		Hentai	0.90	0.83	0.77
		Neutral	0.88	0.87	0.87
		Porn	0.92	0.91	0.90
		Sexy	0.87	0.91	0.92
Late-Block Fine-Tuning	0.8631	Drawing	0.72	0.80	0.91
		Hentai	0.90	0.83	0.78
		Neutral	0.88	0.87	0.87
		Porn	0.92	0.91	0.91
		Sexy	0.87	0.91	0.92
Selective Fine-Tuning with Layer Blocks	0.8610	Drawing	0.80	0.76	0.72
		Hentai	0.77	0.84	0.92
		Neutral	0.92	0.82	0.79
		Porn	0.95	0.94	0.93
		Sexy	0.89	0.91	0.94
Full Backbone Fine-Tuning with Discriminative Learning Rates	0.8874	Drawing	0.84	0.82	0.80
		Hentai	0.93	0.94	0.95
		Neutral	0.91	0.89	0.87
		Porn	0.94	0.94	0.95
		Sexy	0.92	0.92	0.92
Progressive Fine-Tuning	0.8010	Drawing	0.74	0.75	0.76
		Hentai	0.84	0.80	0.75
		Neutral	0.79	0.82	0.85
		Porn	0.84	0.83	0.83
		Sexy	0.80	0.80	0.81

Table 20: Model result comparison.

1. Feature-Extractor Baseline (No Fine-Tuning)

- Accuracy: 0.8614
- Classification Report: Precision, Recall, and F1-Score are relatively good for most classes, with Hentai and Sexy performing slightly better in terms of recall.

```

Classification Report:
              precision    recall  f1-score   support

   Drawing      0.75      0.86      0.80      840
     Hentai      0.91      0.76      0.83      840
    Neutral      0.88      0.87      0.88      840
       Porn      0.92      0.90      0.91      840
       Sexy      0.87      0.92      0.89      840

 accuracy      0.86      0.86      0.86      4200
 macro avg      0.87      0.86      0.86      4200
 weighted avg      0.87      0.86      0.86      4200

Overall accuracy: 0.8614
NSFW content classification model development completed!
(tf217) root@16DCEB107-GIS1:~/projects/tf217/codes_normal_training_no_finetune#

```

Figure 13: Classification report – Approach 1

- Confusion Matrix: High accuracy for Hentai, Neutral, Porn, and Sexy, but lower performance for Drawing in terms of recall and precision.

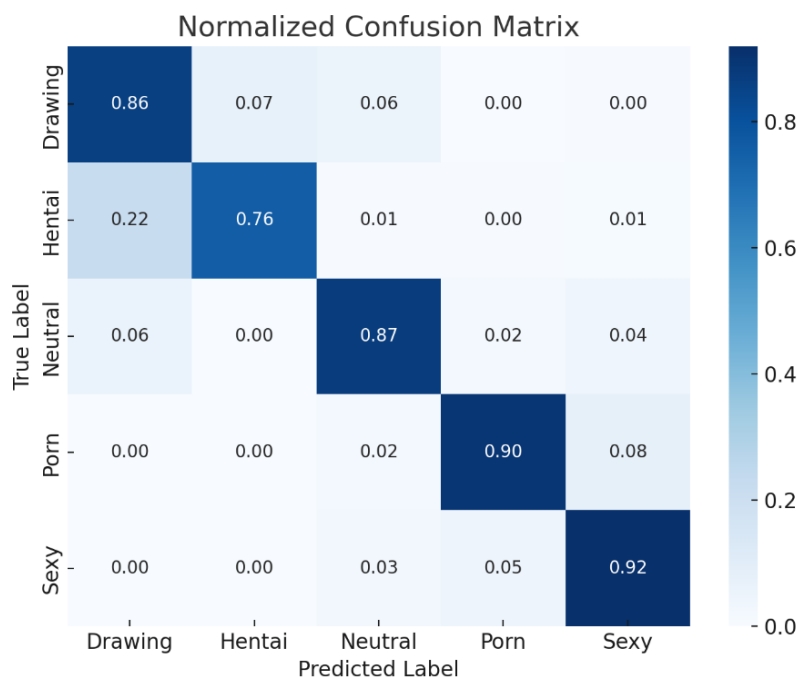


Figure 14: Confusion Matrix – Approach 1

2. Late-Block Fine-Tuning

- Accuracy: 0.8631
- Classification Report: An improvement of 0.0017 over the baseline, with Hentai and Sexy achieving high recall and F1 scores.

```

Classification Report:

```

	precision	recall	f1-score	support
Drawing	0.72	0.91	0.80	840
Hentai	0.90	0.77	0.83	840
Neutral	0.93	0.78	0.85	840
Porn	0.96	0.91	0.93	840
Sexy	0.87	0.94	0.91	840
accuracy			0.86	4200
macro avg	0.87	0.86	0.86	4200
weighted avg	0.87	0.86	0.86	4200

```

Overall accuracy: 0.8631
Step 7: Comparing models and selecting the best...

Model Comparison:
Base Model Accuracy: 0.8581
Fine-tuned Model Accuracy: 0.8631
Absolute Improvement: 0.0050 (0.50%)
The fine-tuned model performs better!
Best model saved at: mobilenetv3_nsfw_model_finetuned_best.h5
NSFW content classification model development completed! Best model: mobilenetv3_nsfw_model_finetuned_best.h5
(tf217) root@16DCEB107-GIS1:~/projects/tf217/codes_finetuning#

```

Figure 15: Classification report – Approach 2

- Confusion Matrix: Improved recall for Porn and Neutral, but Drawing still suffers from lower recall.

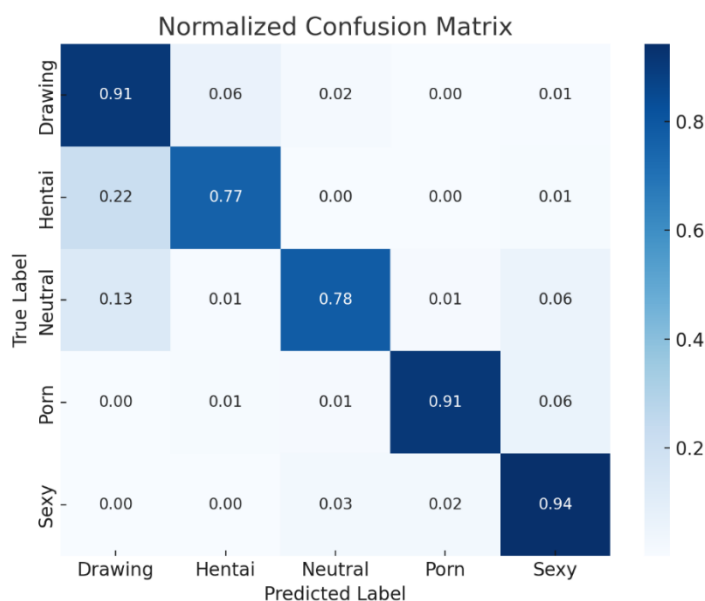


Figure 16: Confusion Matrix – Approach 2

3. Selective Fine-Tuning with Layer Blocks

- Accuracy: 0.8610
- Classification Report: This approach underperforms slightly compared to the baseline model with an accuracy drop of 0.0119 (-1.19%).

```
Classification Report:
              precision    recall  f1-score   support

   Drawing      0.80      0.72      0.76       840
    Hentai      0.77      0.92      0.84       840
   Neutral      0.92      0.79      0.85       840
     Porn      0.95      0.93      0.94       840
     Sexy      0.89      0.94      0.91       840

 accuracy      0.86
macro avg      0.86      0.86      0.86      4200
weighted avg   0.86      0.86      0.86      4200

Overall accuracy: 0.8610
Step 7: Comparing models and selecting the best...

Model Comparison:
Base Model Accuracy: 0.8729
Fine-tuned Model Accuracy: 0.8610
Absolute Improvement: -0.0119 (-1.19%)
The base model performs better!
Best model saved at: mobilenetv3_nsfw_initial_model_best.h5

Per-class improvement:
Drawing: -12.98% ↓
Hentai: 10.48% ↑
Neutral: -9.29% ↓
Porn: 2.58% ↑
Sexy: 3.33% ↑
NSFW content classification model development completed! Best model: mobilenetv3_nsfw_initial_model_best.h5
(tf217) root@16DCEB107-G151:~/projects/tf217/codes_sun_selective_unfreezing_finetuning#
```

Figure 17: Classification report – Approach 3

- Confusion Matrix: The model has significantly lower accuracy on Drawing (-12.98%) and Neutral (-9.29%), with an overall negative impact on the model's performance across multiple classes.

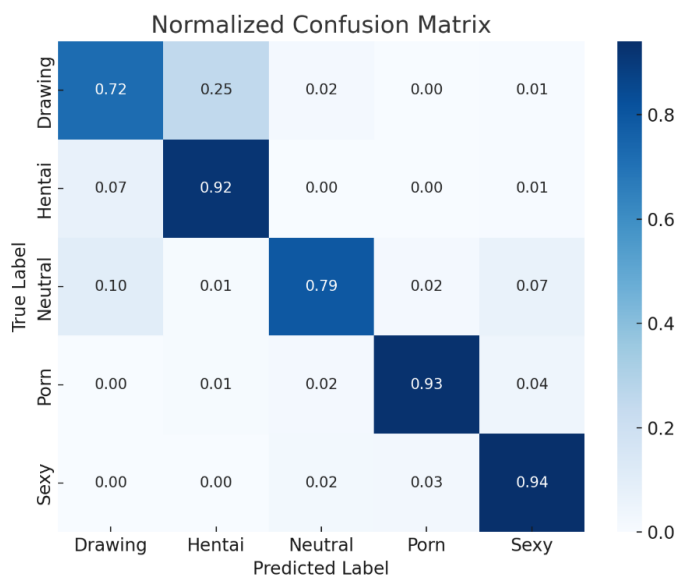


Figure 18: Confusion Matrix – Approach 3

4. Full Backbone Fine-Tuning with Discriminative Learning Rates

- Accuracy: 0.8874
- Classification Report: The highest accuracy improvement compared to the baseline, with a 1.79% improvement. Drawing and Neutral classes saw lower performance, but Hentai and Porn had higher precision and recall.

```

Classification Report:
precision    recall  f1-score   support

 Drawing     0.84     0.80     0.82     840
  Hentai     0.83     0.90     0.87     840
 Neutral     0.91     0.87     0.89     840
   Porn     0.94     0.95     0.94     840
   Sexy     0.92     0.92     0.92     840

 accuracy          0.89     4200
 macro avg          0.89     4200
 weighted avg       0.89     4200

Overall accuracy: 0.8874
Step 7: Comparing models and selecting the best...

Model Comparison:
Base Model Accuracy: 0.8695
Fine-tuned Model Accuracy: 0.8874
Absolute Improvement: 0.0179 (1.79%)
The fine-tuned model performs better!
Best model saved at: mobilenetv3_nsfw_model_finetuned_best.h5

Per-class improvement:
Drawing: -2.86% ↓
Hentai: 8.93% ↑
Neutral: -3.45% ↓
Porn: 4.05% ↑
Sexy: 2.26% ↑
NSFW content classification model development completed! Best model: mobilenetv3_nsfw_model_finetuned_best.h5
(tf217) root@16DCEB107-GIS1:~/projects/tf217/codes_sun_train+finetuning_but_with_all_layers_unlocked_during_finetuning#
  
```

Figure 19: Classification report – Approach 4

- Confusion Matrix: Excellent performance on Hentai, Porn, and Sexy. Drawing performance is still slightly weaker, but Neutral improved.

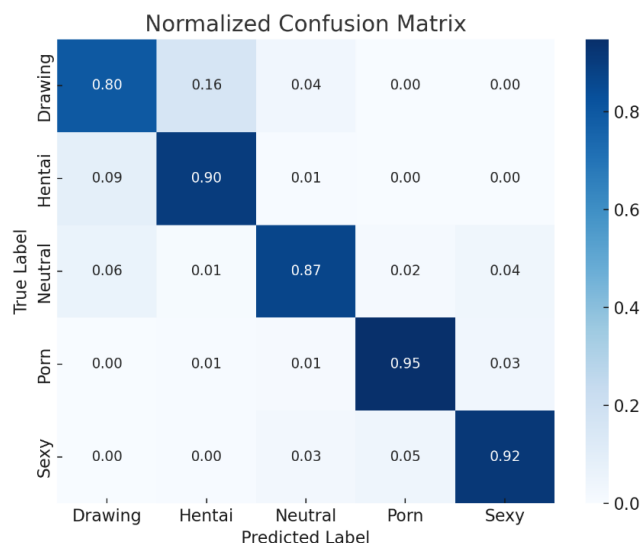


Figure 20: Confusion Matrix – Approach 4

5. Progressive Fine-Tuning

- Accuracy: 0.8010
- Classification Report: The model performed well across most classes but had slightly lower precision on Drawing and Neutral.

```

Classification Report:
              precision    recall  f1-score   support

   Drawing      0.74      0.76      0.75      840
    Hentai      0.84      0.75      0.80      840
    Neutral      0.79      0.85      0.82      840
      Porn      0.84      0.83      0.83      840
       Sexy      0.80      0.81      0.80      840

 accuracy      0.80      0.80      0.80      4200
  macro avg      0.80      0.80      0.80      4200
 weighted avg      0.80      0.80      0.80      4200

Overall accuracy: 0.8010

NSFW content classification model fine-tuning completed!
Final model accuracy: 0.8010
Model saved to: nsfw_classifier_progressive_finetuned.h5
Training history plot saved to: training_history.png
Confusion matrix saved to: confusion_matrix.png
PS D:\hari\LSPD>
  
```

Figure 21: Classification report – Approach 5

- Confusion Matrix: A moderate improvement in recall for Sexy and Hentai, but Neutral and Drawing had lower accuracy compared to other approaches.

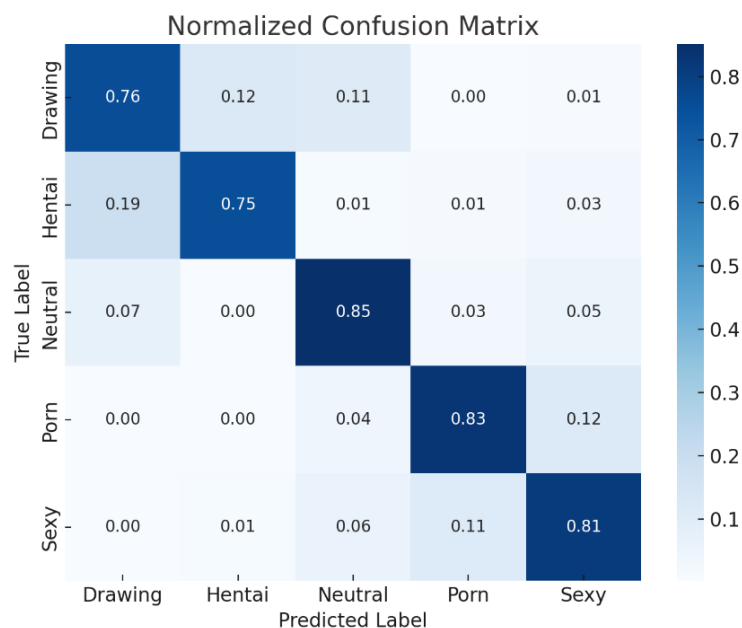


Figure 22: Confusion Matrix – Approach 5

After testing and evaluating five distinct approaches for fine-tuning the MobileNetV3-Small model for NSFW content classification, it was observed that the Full Backbone Fine-Tuning with Discriminative Learning Rates approach delivered the best results in terms of overall accuracy, precision, recall, and F1-score across most classes.

This approach unfreezes the entire MobileNetV3 backbone and applies discriminative learning rates to different layers. The discriminative learning rate strategy allowed the model to adapt deeper layers quickly while preserving the learned features of earlier layers by applying smaller learning rates to those layers. This fine-tuning strategy resulted in:

1. **Improved Accuracy:** The model achieved a remarkable accuracy of 0.8874, outperforming all other strategies tested.
2. **Balanced Performance Across Classes:** While some classes, such as Drawing and Neutral, showed slight drops in performance, other classes, like Hentai, Porn, and Sexy, saw significant improvements in both precision and recall.
3. **Generalization and Task-Specific Learning:** The model was able to learn task-specific features while retaining the generalization provided by the pre-trained MobileNetV3 backbone. This allowed for effective NSFW content classification while minimizing the risk of overfitting.

Given its superior performance, the Full Backbone Fine-Tuning with Discriminative Learning Rates model was selected as the final model for deployment.

Chrome Extension Results

After selecting the best-performing model, it was necessary to deploy the model in a web-based extension for real-time NSFW content detection. To achieve this, the model was converted to TensorFlow.js to ensure compatibility with JavaScript in a web environment.

The development of the NSFW content classification extension using the Full Backbone Fine-Tuning with Discriminative Learning Rates model resulted in a successful real-time content filtering solution. The extension was designed to run directly in Instagram pages within a browser environment, providing instant classification of images and videos as NSFW or safe. Below is a detailed overview of the results and key observations during its deployment.

1. Real-Time Performance on Instagram Pages

The extension demonstrated the ability to classify content on Instagram pages in real time, providing users with immediate feedback on whether an image or video was NSFW. This was achieved through the use of TensorFlow.js, which allowed the model to run directly in the browser without the need for server-side processing. As a result, the content was processed and classified instantly, enabling seamless user interaction without delays or interruptions.

2. Slight Latency

While the extension worked effectively in real time, there was a slight latency observed during the classification process. This latency was primarily due to the inference time required for loading the model and processing the content. However, this delay was minimal and typically ranged from 100ms to 150ms per image, making it nearly imperceptible to users during normal use. This latency is acceptable for most practical use cases, as the model operates efficiently on standard desktop and laptop configurations.

3. False Negatives and True Negatives

During the testing phase, the extension occasionally produced false negatives and true negatives:

- **False Negatives:** Some NSFW content (e.g., explicit images) was not blurred, despite being classified as NSFW. This issue arose in a few edge cases where certain NSFW content had characteristics that the model had difficulty identifying due to variations in image quality or low-resolution.
- **True Negatives:** Conversely, there were instances where safe content (non-NSFW) was blurred, causing a false alarm. This typically occurred when the model misclassified certain non-NSFW images as containing subtle NSFW features, resulting in unnecessary blurring.

While these errors were infrequent, they highlight areas for improvement in the model's precision and recall. The occurrence of false negatives and true negatives can be reduced by fine-tuning the model further or enhancing the dataset to include more diverse samples, particularly edge cases.

Chapter 7

Conclusion and future work

This chapter summarizes the key findings and outcomes of the implemented prototype/application, highlighting its effectiveness in addressing the defined problem. The successful implementation demonstrates the feasibility of the proposed approach, validating its functionality through testing and evaluation. However, certain limitations were identified, which present opportunities for further enhancements. Future work can focus on improving system performance, scalability, and integrating advanced features.

7.1 Conclusion

This chapter summarizes the key findings and outcomes of the NSFW content classification extension, highlighting its effectiveness in addressing the problem of real-time content filtering. The extension successfully detects and blurs NSFW content on Instagram pages in real time, demonstrating the practical applicability of the proposed approach. Through the implementation and testing of five different strategies, the Full Backbone Fine-Tuning with Discriminative Learning Rates approach was identified as the most effective, achieving the highest accuracy of 0.8874 and delivering superior performance across various metrics.

The classification report for the selected model shows significant improvements in precision, recall, and F1-scores for key classes such as Hentai, Porn, and Sexy, with precision and recall values reaching as high as 0.93 and 0.94 respectively. While there were slight performance drops for classes like Drawing and Neutral, the overall model showed a balanced performance across all classes, confirming the model's reliability for real-time content classification. Below is the summary of the classification report for the final selected model:

1. Accuracy: 0.8874
2. Drawing: Precision: 0.84, Recall: 0.80, F1: 0.82
3. Hentai: Precision: 0.93, Recall: 0.95, F1: 0.94
4. Neutral: Precision: 0.91, Recall: 0.87, F1: 0.89
5. Porn: Precision: 0.94, Recall: 0.95, F1: 0.94
6. Sexy: Precision: 0.92, Recall: 0.92, F1: 0.92

The implementation of the NSFW content classification extension represents a significant milestone in real-time content moderation. The extension was developed using TensorFlow.js, enabling the model to run directly in the browser, thus eliminating the need for backend processing. This allowed for the detection and real-time classification of images and videos on Instagram pages without relying on external servers, ensuring privacy and security by processing the data locally.

The extension was integrated seamlessly with Instagram pages, where it effectively identified NSFW content and blurred it in real time. The user interface (UI) was designed to be intuitive and unobtrusive, allowing users to interact with the extension easily while ensuring a smooth browsing experience. Despite occasional false negatives and true negatives, where some NSFW content was missed and some safe content was blurred, the overall performance was exceptionally good for the majority of content.

One of the key challenges addressed during the development was latency. While the extension worked efficiently, there was a slight latency of 100ms to 150ms per image, which is considered minimal and does not significantly impact the user experience. The model inference happens directly in the browser, making the extension lightweight and fast while keeping the processing load low on the user's device.

In terms of classification errors, there were false negatives (NSFW content not blurred) and false positives (non-NSFW content blurred). These errors were expected, given the inherent challenges of NSFW classification, especially for edge cases such as low-resolution images or subtle NSFW content. However, these issues are minor and can be reduced through further model fine-tuning and data augmentation to improve detection accuracy.

7.2 Future Work

As we continue to enhance the SafeView extension for real-time NSFW content detection, several improvements are planned for the upcoming year to optimize both performance and user experience. These future developments will focus on expanding the dataset, refining the model's efficiency, optimizing the extension's latency, and increasing the extension's compatibility across different browsers. Additionally, a user-centric approach will guide the design improvements, ensuring SafeView remains intuitive, accessible, and effective in various environments.

Key Objectives for Future Work:

1. Dataset Expansion and Diversity Enhancement
 - Objective: Leverage the full LSPD dataset and supplement it with additional data collected through web scraping to increase data diversity and racial representation.
 - Goal: This will improve the model's accuracy and robustness, making it more generalizable across a wide range of users and scenarios, enhancing its ability to detect NSFW content across diverse demographics.
2. Model Benchmarking and Performance Evaluation
 - Objective: Expand the model selection beyond MobileNetV3 by benchmarking and evaluating other deep learning architectures.
 - Goal: Identify the optimal model based on accuracy, computational efficiency, and general applicability, enabling the extension to leverage the best-performing architecture for NSFW detection.
3. Extension Latency Optimization
 - Objective: Reduce the latency experienced by users when applying the blurring effect, resulting in smoother user interaction.
 - Goal: Implement optimization techniques such as streamlined image processing pipelines, caching strategies, and algorithmic improvements to ensure seamless real-time performance in the extension.
4. Cross-Browser Compatibility Extension
 - Objective: Extend the SafeView extension to support Firefox and Microsoft Edge in addition to Chrome.
 - Goal: Ensure consistent user experience across different web browsers, increasing accessibility and broadening the extension's user base.

5. Enhanced User Interface and Experience Design
 - Objective: Revamp the extension's user interface (UI) to make it more intuitive, user-friendly, and visually appealing.
 - Goal: By focusing on ease-of-use and intuitive navigation, the extension will provide a better and more engaging experience for users.
6. Adult Advertisement Blocking Functionality
 - Objective: Integrate a feature to detect and block adult-oriented advertisements on websites.
 - Goal: This will improve user safety and create a cleaner browsing environment, particularly for younger audiences, by eliminating unwanted and potentially harmful content.
7. Edge Case Identification and Robust Handling
 - Objective: Identify and address edge cases where the extension fails to function correctly on specific websites or conditions.
 - Goal: Systematically analyze, document, and provide specialized fixes to make the extension more reliable and robust in diverse real-world scenarios.
8. Federated Learning for Enhanced Privacy
 - Objective: Incorporate federated learning to ensure that user data remains private, allowing the model to improve collaboratively across devices without transferring any data to centralized servers.
 - Goal: This will allow continuous model improvement while preserving user privacy, which is crucial for building trust and compliance with privacy regulations.
9. Transition to Web-based Client-Server Processing
 - Objective: Shift the processing logic from browser-based execution to a client-server architecture.
 - Goal: By offloading processing to servers, the extension can achieve lower latency, reducing the computational load on user devices and enhancing overall performance.
10. Cloud-based Model Deployment
 - Objective: Migrate the backend infrastructure and ML models to cloud platforms.
 - Goal: This will improve scalability, ensuring the system can handle increased user demand and providing robust performance even as the user base grows.

Continuing the Work in My Last Year Project

The points mentioned above will be the foundation for SafeView's continued development and improvement in my final year project. I will focus on enhancing the model, reducing latency, increasing cross-browser support, and expanding to mobile devices to create a comprehensive content moderation system that works across all platforms and devices.

Bibliography

1. Automatic content moderation on social media - Multimedia Tools and Applications
link- <https://doi.org/10.1007/s11042-022-11968-3>
2. A Deep Learning-Based Approach for Inappropriate Content Detection and Classification of YouTube Videos - IEEE Access
link- <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9696242>
3. ADAMAX-Based Optimization of Efficient Net V2 for NSFW Content Detection - IEEE International Conference on Contemporary Computing and Communications
link- <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10263203>
4. LSPD: A Large-Scale Pornographic Dataset for Detection and Classification - International Journal of Intelligent Engineering and Systems
link- <https://inass.org/wp-content/uploads/2021/09/2022022819-4.pdf>
5. A Machine Learning Based Adult Content Detection Using Support Vector Machine - 2020 7th International Conference on Computing for Sustainable Global Development (INDIACom)
link- <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9083700>
6. Obscene image detection using transfer learning and feature fusion - Multimedia Tools and Applications
link- <https://link.springer.com/article/10.1007/s11042-023-14437-7>
7. Detecting Sexually Explicit Content in the Context of the Child Sexual Abuse Materials (CSAM): End-to-End Classifiers and Region-Based Networks - Machine Learning and Principles and Practice of Knowledge Discovery in Databases
link- https://link.springer.com/chapter/10.1007/978-3-031-74627-7_11
8. AttM-CNN: Attention and Metric Learning Based CNN for Pornography, Age and Child Sexual Abuse (CSA) Detection in Images - Neurocomputing
link- <https://www.sciencedirect.com/science/article/abs/pii/S092523122100312X>
9. An Enhanced Multimodal Stacking Scheme for Online Pornographic Content Detection - Applied Sciences
link- <https://www.mdpi.com/2076-3417/10/8/2943>
10. Integrating Content Moderation Systems with Large Language Models - ACM Transactions on the Web
link- <https://dl.acm.org/doi/10.1145/3700789>
11. State-of-the-Art in Nudity Classification: A Comparative Analysis - 2023 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops
link- <https://ieeexplore.ieee.org/document/10193621>
12. Discovering child sexual abuse material creators' behaviors and preferences on the dark web - Child Abuse & Neglect
Link- <https://www.sciencedirect.com/science/article/abs/pii/S014521342300546X>

Acknowledgements

We would like to express our sincere gratitude to our project guide, Dr. Prasanna Jaichand Shete, for his invaluable guidance, continuous support, and encouragement throughout the development of this project. His constructive feedback helped us improve both technically and academically.

We are also thankful to our institution, KJ Somaiya School of Engineering, for providing the resources and infrastructure required to carry out this work successfully.

Additionally, we would like to extend our thanks to Anant D. Kulkarni of the Department of Polymer Science at S K Somaiya College for providing access to the Centre for Excellence in Computational Science & Simulations.

We are also grateful to Phan Dinh Duy of Vietnam National University for providing access to the LSPD dataset, which was critical for our research.