**Ain Shams University**
**Faculty of Computer & Information Sciences**
**Computer Science Department**

# Content Guard
## (Content Filter)


**July 2024**


**Ain Shams University**
**Faculty of Computer & Information Sciences**
**Computer Science Department**

# Content Guard
# (Content Filter)

## By:

Andrew Adel Andrews [CS]
Veronica Emad William [CS]
Mehrael Ashraf Rahif [CS]
Monica Sameh Azer [CS]
Mariz Erian Ibrahim [CS]
Filobateer Essam Motamed [CS]

## Under Supervision of:

**Dr. Donia Gamal ElDin**
Computer Science Department,
Faculty of Computer and Information Sciences,
Ain Shams University.

**T.A. Lamis Hassan**
Computer Science Department,
Faculty of Computer and Information Sciences,
Ain Shams University.

# Acknowledgement

We extend our heartfelt gratitude to Dr. Donia Gamal ElDin, whose unwavering support and insightful supervision were pivotal to the success of this project. Her encouragement and challenges inspired us to reach new heights throughout our academic journey. We are equally grateful to TA Lamis Hassan for her invaluable guidance and dedication, consistently pushing us to achieve excellence. Additionally, we offer special thanks to Dr. Salsabeel Amen for her guidance and helpful comments throughout the seminars. Their combined efforts have been instrumental, and we deeply appreciate their contributions. Thank you.

# Abstract

The internet, despite its vast array of resources and benefits, also serves as a conduit for harmful content such as cyberbullying, hate speech, violence, and inappropriate material. This exposure is particularly alarming for vulnerable individuals, including children and adolescents, who may inadvertently come across such detrimental content. Therefore, it is crucial to have effective strategies to ensure a safer online environment. To address this concern, our project focuses on developing a sophisticated real-time content filtering extension that uses advanced machine learning techniques to classify both text and images.

The primary features of our project include:

1. **Text Classification**: The system identifies and filters harmful text content such as cyberbullying, hate speech, and offensive language.
2. **Binary Image Classification**: It distinguishes between violent and non-violent images.
3. **Multiclass Image Classification**: The system categorizes images into specific classes, including fire, accidents, damaged buildings, and normal content.

In developing this system, we curated and prepared diverse datasets for both text and image classification, addressing challenges such as imbalanced data and irrelevant content. Through extensive experimentation with various models, we identified the most effective approaches for each classification task. Specifically, we selected DistilBERT [1] for text classification, MobileNetV3 [2] for binary image classification, and EfficientNetV2B2 [2] for multiclass image classification.

The results of our developed system have demonstrated its ability to effectively filter harmful content in real time, thus significantly enhancing the safety and inclusivity of the online environment. By proactively identifying and mitigating the risks associated with harmful content, our project contributes to fostering a positive and secure digital space for all users. This, in turn, promotes a healthier and more supportive online community, free from the detrimental impacts of cyberbullying, hate speech, and exposure to violent or inappropriate material.

# الخلاصة

على الرغم من الموارد والفوائد الهائلة التي يوفرها الإنترنت، إلا أنه يعمل أيضًا كقناة للمحتوى الضار مثل التنمر الإلكتروني وخطاب الكراهية والعنف والمواد غير اللائقة. يثير هذا التعرض قلقًا خاصًا للأفراد الأكثر ضعفًا، بما في ذلك الأطفال والمراهقين، الذين قد يتعرضون لمثل هذا المحتوى الضار عن غير قصد. لذلك، من الضروري وجود استراتيجيات فعالة لضمان بيئة آمنة عبر الإنترنت.

**يهدف مشروعنا إلى تطوير ملحق متطور لتصفية المحتوى في الوقت الفعلي يستخدم تقنيات متقدمة للتعلم الآلي لتصنيف النصوص والصور.**

الميزات الرئيسية لمشروعنا تشمل:

1. **تصنيف النص:** يقوم النظام بتحديد وفلترة محتوى النص الضار مثل التنمر الإلكتروني وخطاب الكراهية واللغة المسيئة.
2. **تصنيف الصور الثنائي:** يميز بين الصور العنيفة وغير العنيفة.
3. **تصنيف الصور متعدد الفئات:** يقوم النظام بتصنيف الصور إلى فئات محددة، بما في ذلك الحرائق والحوادث والمباني المتضررة والمحتوى العادي.

في تطوير هذا النظام، قمنا بتنظيم وإعداد مجموعات بيانات متنوعة لتصنيف النصوص والصور، مع معالجة تحديات مثل البيانات غير المتوازنة والمحتوى غير ذي الصلة. من خلال تجربة مكثفة مع نماذج مختلفة، حددنا أكثر الطرق فعالية لكل مهمة تصنيف. على وجه التحديد، اخترنا DistilBERT [1] لتصنيف النصوص وMobileNetV3 [2] لتصنيف الصور الثنائي وEfficientNetV2B2 [3] لتصنيف الصور متعدد الفئات.

أظهرت النتائج النهائية لنظامنا المطور قدرته على تصفية المحتوى الضار بفعالية في الوقت الفعلي، مما يعزز بشكل كبيرمن سلامة وشمولية البيئة عبر الإنترنت. من خلال تحديد المخاطر المرتبطة بالمحتوى الضار والتخفيف منها بشكل استباقي، يساهم مشروعنا في تعزيز مساحة رقمية إيجابية وآمنة لجميع المستخدمين.
وهذا بدوره يعزز مجتمعًا عبر الإنترنت أكثر صحة ودعمًا وخالٍ من الآثار الضارة للتنمر الإلكتروني وخطاب الكراهية والتعرض للمواد العنيفة أو غير اللائقة.

# Table of Contents

# List of Figures

# List of Tables

# List of Abbreviations

| Abbreviation | What the abbreviation stands for |
|---|---|
| API | Application Programming Interface |
| AI | Artificial Intelligence |
| CSS | Cascading Style Sheets |
| CNBD | Combinational Network for Bullying Detection |
| CNN | Convolutional neural network |
| HTML | Hypertext Markup Language |
| HTTP | Hypertext Transfer Protocol |
| JS | JavaScript |
| KNN | K-Nearest Neighbors |
| ML | Machine Learning |
| MNB | Multinomial Naive Bayes |
| NLP | Natural Language Processing |
| PIL | Python Imaging Library |
| ResNet | Residual Network |
| SGD | Stochastic Gradient Descent optimizer |
| SVC | Support Vector Classifier |
| SVM | Support Vector Machine |
| VGG | Visual Geometry Group |

# 1- Introduction

## 1.1 Motivation

Our motivation stems from the increasing prevalence of harmful online content, particularly hateful speech and violence, on social media and educational platforms.

This content negatively impacts user's well-being, highlighting the urgent need for a safer online environment. We are dedicated to creating a secure space for vulnerable users, ensuring they are not exposed to inappropriate and violent material.

## 1.2 Problem Definition

Despite the internet's essential role in daily life, it also harbors harmful content, especially on social media platforms where inappropriate and violent material is prevalent.

Although some graphic content is filtered, many posts containing violent imagery merely receive warnings and remain accessible.

Additionally, curse words often evade removal by being obscured with various symbols.

The challenge, therefore, is to develop an effective filtering system to ensure a safe and positive online experience for all users.

## 1.3 Objective

The objective of this project is to develop an effective filtering system that enhances the safety and positivity of online experiences by thoroughly identifying and removing harmful content, including inappropriate, violent imagery, and obscured curse words.

This system is not restricted to a specific platform and aims to protect vulnerable users, especially children, from exposure to such content, thereby contributing to their well-being and creating a safer online environment.

Additionally, we aim to create a background tool that filters text and images using advanced machine learning for real-time application, addressing biases to ensure fair treatment of diverse perspectives.

## 1.4 Time Plan

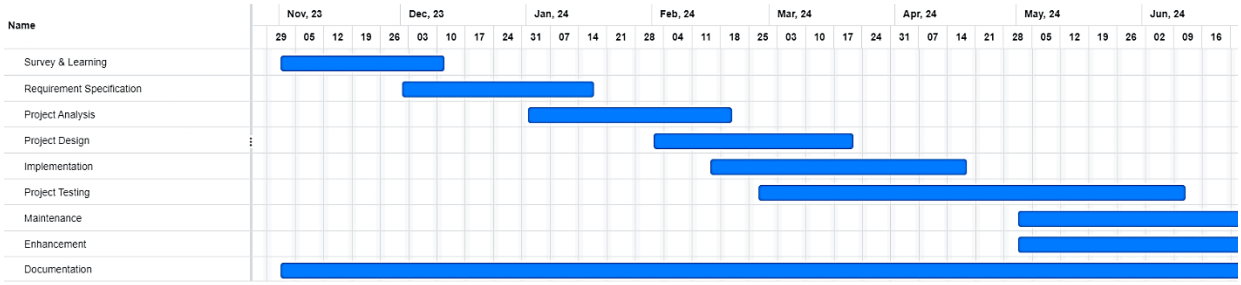


*Figure 1: Time Plan*

**Figure 1** presents the overall project timeline, highlighting the main stages and their respective durations. The project begins with the literature review and progresses through to the final product development.
As shown, some tasks overlap.
Finally, the project documentation is an ongoing activity throughout the project.

## 1.5 Document Organization

**Chapter 2**

This chapter provides an overview of the project's background, focusing on its scientific foundation and intended application area. It also includes a brief description of other similar projects.

**Chapter 3**

This chapter details the system architecture, explaining each component and the deep learning models used. It also describes the system's intended users and the basic knowledge they need to use the project effectively.

**Chapter 4**

This chapter gives an in-depth explanation of each function, technique, and algorithm implemented in the project, along with the testing procedures performed.

**Chapter 5**

This chapter offers a comprehensive user manual, including a step-by-step guide with screenshots on how to install and use the system.

**Chapter 6**

This chapter summarizes the documentation and suggests future improvements to enhance the project's potential.

# 2- Background

## 2.1 Description of the field

**Insufficiency of Filtering Guidelines on Popular Social Media Platforms:**

Despite the established filtering guidelines on popular social media platforms, such as Facebook, Twitter, and Instagram, they often fall short of effectively filtering out crucial harmful content. This includes graphic violence, hate speech, and other forms of inappropriate material that can evade detection or receive minimal repercussions.

**Limitations in Natural Language Processing (NLP) for Contextual Understanding:**

In Natural Language Processing (NLP), current filtering methods often rely on identifying explicit keywords, such as curse words, to flag potentially harmful content. These methods struggle to understand the nuanced context of a hateful comment that may not explicitly contain offensive language. As a result, harmful content lacking explicit triggers can go unnoticed or unaddressed.

**Lack of Filtering Systems on Other Platforms:**

Many existing platforms outside the major social media networks do not have robust content filtering systems implemented. This gap leaves users vulnerable to encountering harmful content without adequate protection or moderation measures in place.

**Issues of Fair Treatment in Content Filtering**:

There is a significant issue regarding fairness in how content filtering systems operate. Biases in algorithms or human moderation can lead to uneven treatment of different perspectives or demographic groups. Certain voices may be disproportionately silenced or penalized, while others may evade scrutiny despite violating similar guidelines.

**Targeting Unsupervised Children as Users**:

One of the key user groups targeted by this project is unsupervised children who browse the web. These young users are particularly vulnerable to encountering harmful content due to their limited understanding of online risks and the potential for content filtering systems to fail to protect them adequately.

## 2.2 Scientific Background

**Harmful Content Definition:**

Content that may cause emotional, psychological, or bodily harm to people.
This includes text and images containing threats, insults, offensive language, or any form of aggressive behavior online.

**Harmful Content Categories Covered:**

**Text** [16]

- Aggression
- Cyberbullying
- Hate Speech
- Offensive Language
- Normal

**Image** [9]

• Violence
• Fire
• Accidents
• Damaged Buildings
• Normal

**Traditional Machine Learning Approaches:**

Traditional machine-learning approaches offer simplicity, efficiency, and interpretability, making them suitable for certain applications, especially with limited data and computational resources.

They often fall short in handling complex, large-scale, and context-rich data compared to deep learning models, which can automatically learn intricate patterns and features from the data.

**Deep Learning Approaches:**

Deep learning approaches offer significant advantages in terms of automatic feature extraction, contextual understanding, and performance on large-scale and complex datasets for both text and image classification. These benefits come with challenges, including high computational requirements, large data needs, and difficulties in interpretability.

## 2.3 Literature Review

### 2.3.1 Overview:

The project aims to develop a real-time content filtering extension that leverages machine learning to classify text and images, effectively filtering harmful content like cyberbullying, hate speech, violence, and inappropriate material to create a safer online environment. It utilizes DistilBERT[1]for text classification, MobileNetV3[2] for binary image classification (violent/non-violent), and EfficientNetV2B2[3]for multiclass image classification (fire, accidents, damaged buildings, normal content). The system is designed to be user-friendly, requiring no technical expertise, and is intended for various users, including parents, educational institutions, content creators, and mental health organizations.

### 2.3.2 Related Work:

The related work primarily discusses the content filtering and moderation efforts of major social media platforms like Facebook and Twitter. These platforms employ a combination of automated systems, utilizing machine learning algorithms and natural language processing (NLP) techniques, along with human moderators to review flagged content. The related work also mentions Instagram and X (formerly Twitter) and their content guidelines, highlighting the challenges these platforms face in enforcing their policies due to the vast amount of user-generated content.

In contrast, this project focuses on developing a browser extension for real-time content filtering. While the project draws inspiration from the techniques employed by social media platforms, it aims to provide a more personalized and user-controlled filtering experience. The browser extension will leverage machine learning models and rule-based filtering to empower users to tailor their content preferences and block harmful content according to their individual needs. Additionally, the project emphasizes addressing biases in content filtering by utilizing diverse datasets and incorporating user feedback, which is a distinct aspect not explicitly mentioned in the related work concerning social media platforms.

| Platform Name | Guidelines | Observations From previous experience |
|---|---|---|
| **Instagram [21]** | • Intellectual Property<br>• Appropriate Imagery<br>• Spam<br>• Illegal Content<br>• Hate Speech, Bullying and Abuse<br>• Self-Injury<br>• Graphic Violence | • Even though Graphic content is filtered out there are posts that contain graphic and violence imagery that just get a warning and can be viewed.<br><br>• There are curse words written in a lot of posts and separated by a variety of symbols that don't get removed |
| **Facebook [22]** | • Violence and Criminal Behavior<br>• Objectionable content<br>• Integrity and authenticity | |
| **X (Twitter) [23]** | • Hateful references<br>• Incitement<br>• Slurs and Tropes<br>• Dehumanization<br>• Hateful Imagery<br>• Hateful Profile | |

*Table 1: Social media platforms guidelines*

Table 1 illustrates a comparison of content guidelines across three major social media platforms: Instagram, Facebook, and X (formerly Twitter). Each platform has established specific rules and restrictions regarding the types of content allowed on their sites.

**Instagram:** Instagram's guidelines focus on intellectual property, appropriate imagery, spam, illegal content, hate speech, bullying, abuse, self-injury, and graphic violence. The platform aims to maintain a positive and safe environment for its users by prohibiting harmful and offensive content [21].

**Facebook:** Facebook's guidelines address a broader range of issues, including violence, criminal behavior, objectionable content, and maintaining integrity and authenticity. The platform strives to create a space where users can connect and share experiences while upholding community standards [22].

**X (Twitter):** X's guidelines specifically target hateful references, incitement, slurs, tropes, dehumanization, and hateful imagery, including profile

information. The platform is committed to combating hate speech and promoting healthy online discourse [23].

While all three platforms share the goal of creating a safe and respectful online environment, their guidelines differ in scope and emphasis. Instagram focuses on protecting users from harmful content, Facebook emphasizes community standards and authenticity, and X prioritizes combating hate speech and promoting positive interactions.

| Authors | Dataset | Focus of paper | Models used | Results |
|---|---|---|---|---|
| Rubio, J. L. S., Almeida, A. V., & Segura-Bedmar, I. (2023) [9]. | dataset comprising 2,996 Spanish tweets related to violent acts | Detection of Violent Incidents on Social Networks (Text) | BERT | 0.918 F1-score |
| Escalante-Hernandez, A., Joaquín-Arellano, L., Lavalle-Martínez, J. D. J., Villaseñor-Pineda, L., & Jair Escalante, H. (2023) [14] | DA-VINCIS dataset of 5000 tweets in Mexican and Spanish | Identification of violent incidents using social networks (Text) | MLP | 0.65 F1-Score |
| Pericherla, S., & Ilavarasan, E. (2023) [7]. | MS-COCO, Flickr30K | Cyberbullying detection in image (images) | RoBERTa | Accuracy 88.82% |

*Table 2: Related work table*

Table 2 presents a concise overview of three distinct research papers, each employing different machine learning models to address the challenge of content filtering.

- The first paper utilizes a Decision Tree model for text classification, achieving a remarkable accuracy of 99.90%. Decision Trees are known for their interpretability and ability to handle both categorical and numerical data, making them a suitable choice for this task [9].
- The second paper employs a Multinomial Naive Bayes (MNB) model, also for text classification, but with a lower accuracy of 64.00%. MNB is a probabilistic model that assumes feature independence, which may not

always hold true in real-world text data, potentially explaining the lower accuracy [14].

- The third paper focuses on image classification and utilizes a Medium Gaussian SVM model, achieving an accuracy of 98.73%. SVMs are effective in high-dimensional spaces and can handle non-linear relationships between features, making them well-suited for image analysis [7].

This table underscores the diversity of approaches and models employed in content filtering research, highlighting the ongoing exploration of various techniques to enhance accuracy and effectiveness in identifying and classifying harmful content.

| # | **Model Name** | **Accuracy** | **Image / text** | **Reference** |
|---|---|---|---|---|
| 1 | Decision Tree | 99.90% | Text | [4] |
| 2 | Multinomial Naïve Bayes (MNB) | 64.00% | Text | [4] |
| 3 | AdaBoost Classifier | 45.50% | Text | [4] |
| 4 | Logistic regression | 77.70% | Text | [4] |
| 5 | Linear SVC | 97.00% | Text | [4] |
| 6 | Stochastic gradient descent optimizer (SGD) | 92.80% | Text | [4] |
| 7 | Random Forest | 99.00% | Text | [4] |
| 8 | Bagging classifier | 93.80% | Text | [4] |
| 9 | Medium Gaussian SVM | 98.73% | Image | [11] |
| 10 | KNN | 82.26% | Image | [11] |
| 11 | Logistic regression & SVM | 75.00% | Text | [16] |
| 12 | decision tree, SVM & Naive | 91.00% | Text | [16] |

| | | | | |
|---|---|---|---|---|
| 13 | Combinational Network for Bullying Detection (CNBD) | 91.40% | Image | [7] |

*Table 3: Models used in previous work*

Table 3 presents a list of machine learning models used in previous research for text and image classification tasks, along with their reported accuracy.

- Models 1-8 are designed for text classification, while models 9-10 are for image classification.
- Model 11 combines logistic regression and SVM for text classification, while model 12 combines decision trees, SVM, and Naive Bayes for the same purpose.
- Model 13, the Combinational Network for Bullying Detection (CNBD), is specifically designed for image-based bullying detection.

The table highlights the diversity of approaches used in the field, with both traditional machine learning methods (like decision trees and Naive Bayes) and more complex models (like CNBD) being employed. The accuracy rates vary widely, suggesting that the choice of model and its performance can be highly dependent on the specific task and dataset.

### 2.3.3 Similar Systems:

**HaramBlur Extension:**

HaramBlur is a browser extension designed to filter out content that may be considered inappropriate or offensive according to Islamic values. It also aims to protect user privacy and reduce browsing distractions. The extension automatically detects, and blurs content deemed "Haram", allowing users to navigate the web while adhering to their religious beliefs [24].

While both HaramBlur and our project share the goal of content filtering, there are key differences in their scope and target audience. HaramBlur is specifically

tailored for users who wish to filter content based on Islamic values, whereas our project aims to create a more general-purpose content filter that can be customized to individual preferences. Additionally, HaramBlur focuses on blurring inappropriate content, while our project aims to block or remove it entirely.

Another distinction lies in the type of content being filtered. HaramBlur primarily targets images and videos, while our project encompasses both text and image filtering. This broader scope allows our project to address a wider range of harmful content, including cyberbullying, hate speech, and violence material.

# 3- Analysis and Design

## 3.1 System Overview

### 3.1.1 System Architecture

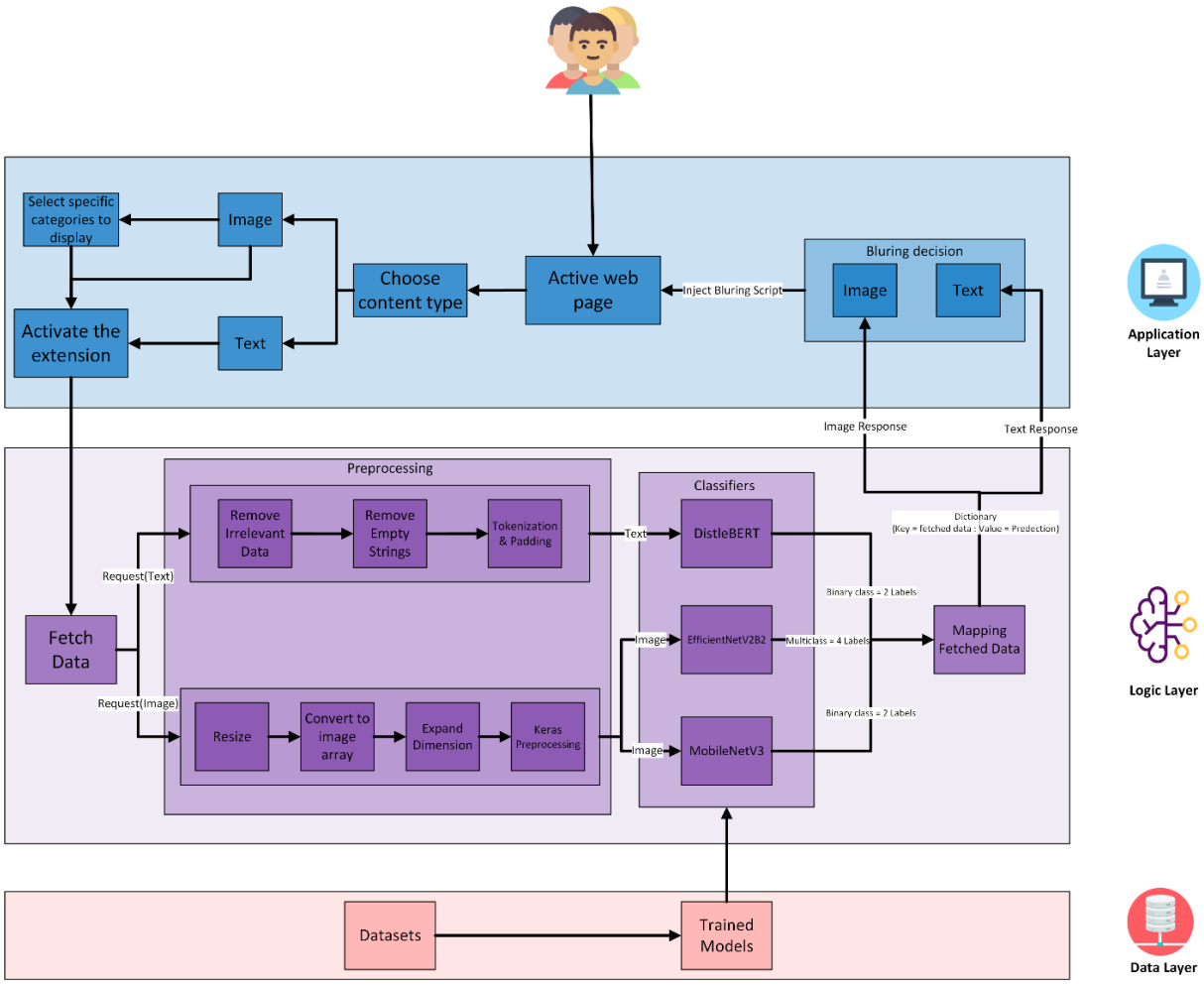


*Figure 2: System Architecture*

### 3.1.2 Description of Methods and Procedures

**1. Application Layer:**

- User Interface and communication layer where the end-user interacts with the application.

- Functionalities:

1) Choose Content-Type: The user selects the type of content to be filtered.

2) Select Specific Categories to display: The user can choose specific image categories to whitelist and bypass the filtering.

3) Activate the Extension: The user initiates the extension to start filtering the content on the current active web page.

4) Blurring Decision: The application injects a blurring script based on the prediction of the models, to blur text or images that match the user's filtering criteria.

**2. Logic Layer:**

- Processes the data collected from the Application Layer and applies the trained models to get the filtering decision.
- Functionalities:

1) Preprocessing:
   - Images: Converts to image arrays, resizes, and expands dimensions as needed.
   - Text: Removes irrelevant data, eliminates empty strings, performs tokenization, and adds padding.

2) Classifiers:
   - Image classification:
     - EfficientNet V2 B2 model to classify images into 4 classes: Fire, Damaged Building, Accidents, and Normal.
     - MobileNet V3 model to classify images into 2 classes: Violent and Non-violent.
   - Text classification:
     - DistilBERT model to analyze text and identify inappropriate language and topics.

3) Mapping Fetched Data:
   - Maps each data item to its filtering decision.

**3. Data Layer:**

- Stores the datasets used to train the deep learning models.
- Provides an interface to access and load the trained models during operation.

## 3.1. System Users

*A. Intended Users:*

Our intended user is anyone that browses the internet daily that faces a lot of undesired content and needs help to filter out unwanted content.

Such as:

1. **Parents and Guardians:**
    - Parents who want to protect their children from inappropriate or harmful content online.
    - Guardians oversee the online activities of minors to ensure a safe browsing environment.

2. **Educational Institutions:**
    - Schools and universities to ensure students are not exposed to harmful content while using school-provided internet and devices.
    - Libraries and other educational facilities to provide a safe browsing environment for all users.

3. **Content Creators, Moderators and Social Media Users:**
    - Safeguard users from offensive and harmful content.
    - Bloggers, vloggers, and online communities want to ensure their content remains clean and appropriate for all audiences.
    - Moderators of forums and online communities who need tools to help maintain a safe and respectful environment.

4. **Mental Health, Support Groups and Healthcare Providers:**
    - Prevent exposure to triggering or harmful content in online support communities.
    - Clinics and hospitals that want to protect patients, especially those in vulnerable mental states, from exposure to triggering content.

### *B. User Characteristics*

One of our distinct features is that the user does not need to have any technical skill or technical background to be able to operate the extension since it has a user-friendly interface that makes it accessible for all ages and tech proficiency levels.

## 3.2 System Analysis & Design

### 3.2.1 Use Case Diagram

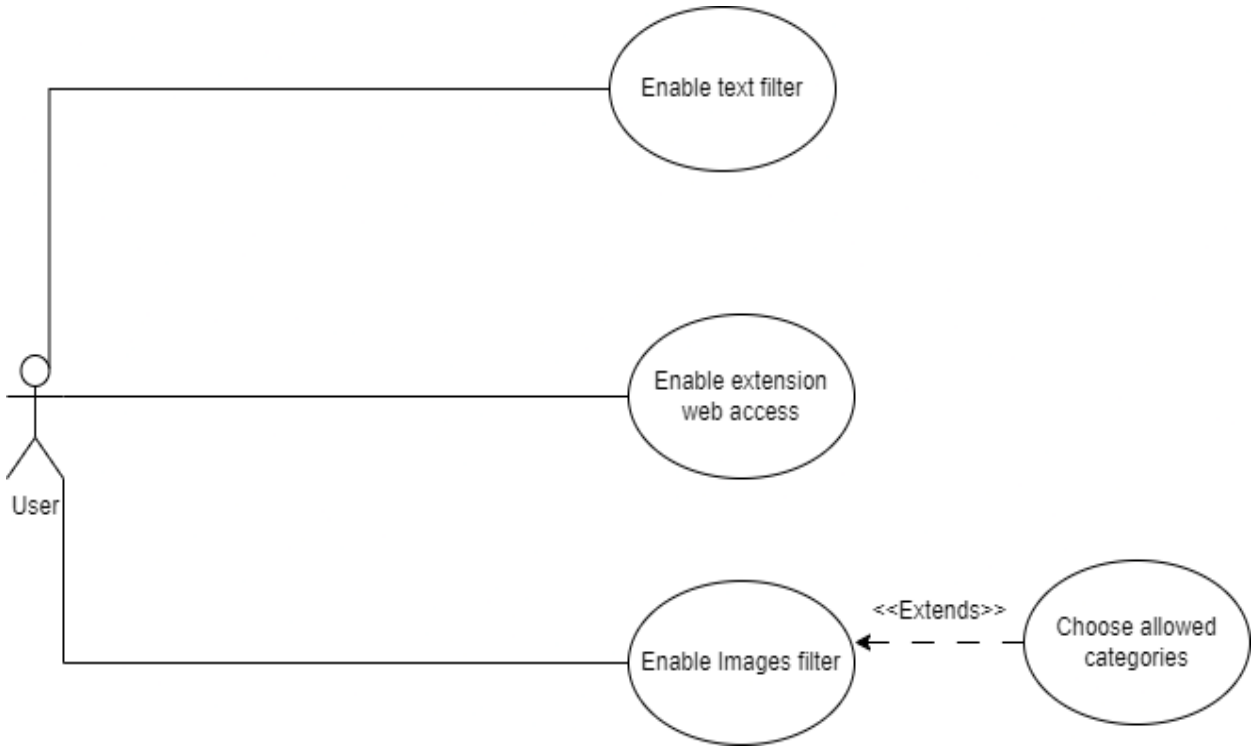


*Figure 3: Use Case Diagram*

1. **Enable Extension Web Access:**
    - **Description:** The user grants the browser extension permission to access and interact with the content of web pages. This is a fundamental step, as it allows the extension to analyze and filter content.
    - **Actor:** User
    - **Trigger:** User activates the extension.
    - **Precondition:** The extension is installed but not yet granted web access.
    - **Postcondition:** The extension can now scan and process web page content.

2. **Enable Text Filter:**
    - **Description:** The user activates the text filtering feature of the extension. This enables the analysis and potential blurring of text content based on predefined criteria.
    - **Actor:** User
    - **Trigger:** User toggles the text filter setting within the extension's interface.
    - **Precondition:** The extension is installed.
    - **Postcondition:** The extension will scan and filter text content on current active web page.

3. **Enable Images Filter:**
    - **Description:** The user activates the image filtering feature of the extension. This enables the analysis and potential blurring of images based on predefined criteria.
    - **Actor:** User
    - **Trigger:** User toggles the image filter setting within the extension's interface.
    - **Precondition:** The extension is installed.
    - **Postcondition:** The extension will scan and filter images on current active web page

4. **Choose Allowed Categories:**
    - **Description:** This use case is an extension of "Enable Images Filter." It allows the user to further refine the image filtering process by selecting specific categories of images that they want to allow (i.e., not be blurred).
    - **Actor:** User
    - **Trigger:** User clicks on settings button to customize the allowed image categories.
    - **Precondition:** Image filtering must be enabled.
    - **Postcondition:** The extension will only filter images that do not fall into the allowed categories selected by the user.

**Relationship Between Use Cases:**

- The "Choose Allowed Categories" use case extends the "Enable Images Filter" use case. This means it is an optional, additional step that the user can take after enabling image filtering. The "extends" relationship indicates that this use case is only relevant or available under certain conditions (i.e., when image filtering is active).
- Both "Enable Text Filter" and "Enable Images Filter" can be performed independently.

## 3.2.2 Class Diagram

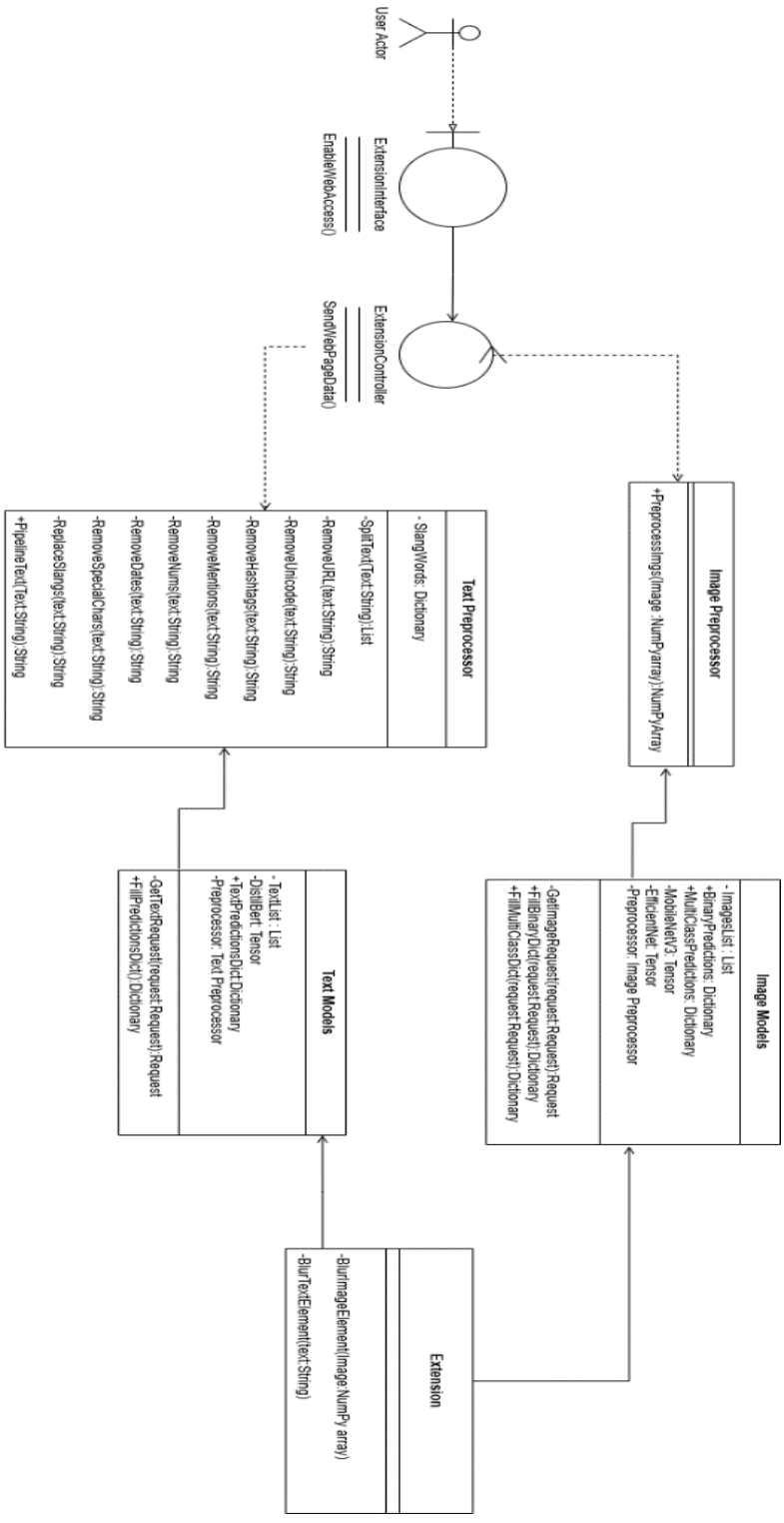


Figure 4: Class Diagram

**Core Processing Classes:**

- **Image Preprocessor:**
  - **Purpose:** Takes raw image data (NumPyArray) and transforms it into a format suitable for the image models.
  - **Key Methods:** PreprocessImg(Image: NumPyArray)

- **Image Models:**
  - **Purpose:** Houses different deep learning models for image analysis (MobileNetV3, EfficientNet).
  - **Key Attributes:**
    - BinaryPredictions: Dictionary of binary image classifications (e.g., "Violence"/"not Violence ").
    - MultiClassPredictions: Dictionary of multi-class image classifications (e.g., "Fire," "Accident," "Damaged Building").
  - **Key Methods:**
    - GetImageRequest(request: Request): Takes a user request and processes it for image classification.
    - FillBinaryDict/FillMultiClassDict(request: Request): Returns prediction results based on the request type.

- **Text Preprocessor:**
  - **Purpose:** Cleans and prepares raw text data for analysis by the text models.
  - **Key Attributes:**
    - StopWords: Dictionary of words to exclude from analysis.
  - **Key Methods:** A variety of text cleaning functions like SplitText, RemoveURL, RemoveUnicode, etc.

- **Text Models:**
  - **Purpose:** Utilizes deep learning models for text analysis (DistilBERT in this case).
  - **Key Attributes:**
    - TextList: List of processed text segments.
    - TextPredictionsDict: Dictionary storing text analysis results.

- **Key Methods:**
  - GetTextRequest(request: Request): Takes a user request and prepares it for text analysis.
  - FillPredictionsDict(): Returns the results of the text analysis.

**Interface and User Interaction:**

- **ExtensionInterface:**
  - **Purpose:** Defines the contract (methods) that extensions must implement to interact with the system.
  - **Key Method:** EnableWebAccess(): Used to enable communication between the extension and the core system.

- **ExtensionController:**
  - **Purpose:** Acts as a bridge between the user/extension and the core processing classes.
  - **Key Method:** SendWebPageData(): Sends processed data back to the user interface.
- **Extension:**
  - **Purpose:** Represents the user interface or any external tool interacting with the system.

  - **Key Methods:**
    - BlurImageElement(Image: NumPyArray): Applies blurring to an image.
    - BlurTextElement(text: String): Applies blurring or obfuscation to text.

**User Actor:**

- **Purpose:** Represents the person interacting with the system.

**Key Relationships:**

- The User Actor communicates with the system through the Extension.
- The ExtensionController manages communication between the Extension and the core processing classes.
- The Image/Text Preprocessors prepare data for the Image/Text Models.
- The Image/Text Models produce predictions stored in dictionaries.

## 3.2.3 Sequence Diagram

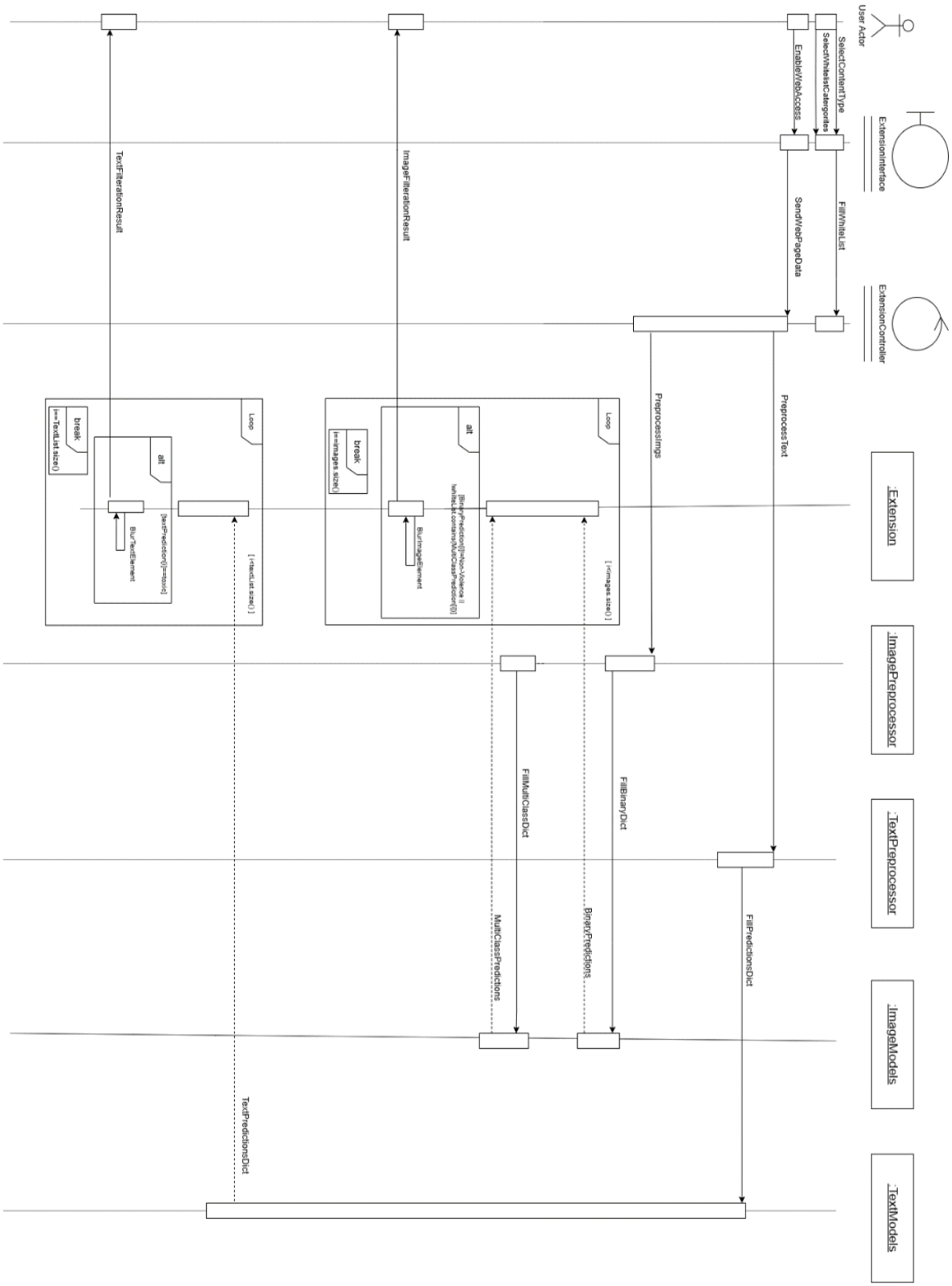


*Figure 5: Sequence Diagram*

Main interactions and processes involved in content filtering within the Extension system.

1. **User Interaction and Setup**:
    - The diagram starts with the User Actor interacting with the Extension Interface.
    - The user selects the Content Type (either image or text) and the Filter Categories to be applied.
    - The ExtensionController sends a signal to Enable Web Access and then sends the Web Page Data to the Extension Interface.

2. **Image Filtering Process**:
    - If the selected content type is an image, the Image Preprocessor is called upon to Preprocess the image data.
    - The preprocessed image is then passed to the Image Models for analysis and prediction, resulting in a Fill Prediction Dict.
    - The Image Models then perform binary and multi-class predictions, stored in Binary Dict and MultiClass Dict, respectively.
    - The Image Filtering Result is determined and sent back to the Extension Interface.

3. **Text Filtering Process**:
    - If the selected content type is text, the Text Preprocessor is responsible for Preprocessing the text data.
    - The preprocessed text is then fed into the Text Models for analysis and prediction, resulting in a Text Predictions Dict.
    - The Text Models perform binary predictions, and the result is stored in the Binary Predictions.

- The Text Filtering Result is determined based on these predictions and sent back to the Extension Interface.

4. **Result Presentation**:
    - The Extension Interface receives the filtering results (either Image Filtration Result or Text Filtration Result).
    - The results are then displayed to the user. The diagram indicates this step with a return value of either "Image Filtered" or "Text Filtered" to the User Actor.

# 4- Implementation and Testing

## 4.1 Datasets

### 4.1.1 Text Datasets

Our dataset is merged from 5 different datasets.

| Category | Datasets | Number of used records | Our Dataset |
|---|---|---|---|
| **Inappropriate content** | Cyberbullying dataset (aggression) [27] | 14782 records | **167,051** records |
| | Cyberbullying dataset (Kaggle) [27] | 2806 records | |
| | Tweets 1 [16] | 20620 records | |
| | Toxic comments detection (Balanced) [28] | 89363 records | |
| | Toxic comments detection (Classified comments) [28] | 39480 records | |
| **Appropriate content** | Cyberbullying dataset (aggression) [27] | 101082 records | **165,669** records |
| | Cyberbullying dataset (Kaggle) [27] | 5993 records | |
| | Tweets 1 [16] | 4163 records | |
| | Toxic comments detection (Balanced) [28] | 100513 records | |
| | Toxic comments detection (Classified comments) [28] | 102425 records | |

*Table 4: Text Dataset*

**Challenges:**

- Some datasets contained abbreviated slang words (lol, asap).
- It also contained irrelevant content such as days, months, mentions, links, etc.
- Most of the datasets were imbalanced.

**How we solved them:**

- Searched for a dataset that contains the original words of the slang word abbreviation.
    - Created a dictionary to replace them with their original words.
    - Ex: lol = laughing out loud, asap = as soon as possible.

- Removed irrelevant content by
  - Creating a list of short and long versions of each word.
  - Ex: Mon = Monday, Apr = April.
  - Used regular expressions to remove these special characters.
- Applied down sampling as a balancing technique.

### 4.1.2 Image Datasets

#### *4.1.2.1 Binary Image Dataset:*

Our dataset is merged from two different datasets.

| Category | Datasets | Number of used images | Our Dataset |
|---|---|---|---|
| **Violence** | Real Life Violence and Non-Violence Data [25] | 5832 images | **11,674** images |
| | Violence vs. Non-Violence: 11K Images Dataset [26] | 5842 images | |
| **Non-Violence** | Real Life Violence and Non-Violence Data [25] | 5231 images | **10,462** images |
| | Violence vs. Non-Violence: 11K Images Dataset [26] | 5231 images | |

*Table 5: Binary Image Classification Dataset*

Both datasets were clean and devoid of significant issues.
However, to enhance the deep learning model's performance, the decision was made to merge them.
By combining the datasets, we capitalized on their strengths, resulting in a more diverse and effective training set.

#### *4.1.2.2 Multiclass Image Dataset:*

Our dataset is merged from 12 different datasets as each category is a combination of selected folders from 3-5 different datasets.

| Category | Dataset | Number of used images | Our Dataset |
|---|---|---|---|
| **Fire** | Cyclone Wildfire Flood Earthquake Database [33] | 1077 images | **16,289** images |
| | Forest Fire [30] | 1022 images | |
| | The wildfire dataset V2 [32] | 730 images | |
| | Forest_Fire_Smoke_and_Non_Fire_Image_Dataset [35] | 10800 images | |
| **Normal** | Wildfire_surveillance | 2580 images | **17,098** images |
| | The wildfire dataset V2 [32] | 1157 images | |
| | Fire Images Database [37] | 3952 images | |
| | FOREST FIRE IMAGE DATASET [34] | 7139 images | |
| | Wildfire_surveillance [32] | 2580 images | |
| **Accidents** | AIDER [29] | 485 images | **16,011** images |
| | Accident detection | 1675 images | |
| | Augmented image from (Traffic-Net dataset) | 13850 images | |
| **Damaged Buildings** | RescueNet [38] | 3595 images | **15,250** images |
| | Cyclone Wildfire Flood Earthquake Database [33] | 1350 images | |
| | AIDER [29] | 511 images | |
| | Augmented Images from all previous datasets in this category | 4164 images | |
| | Extracted buildings images from (Ukraine Images 2023) [31] | 5630 images | |

*Table 6: Multiclass Image Classification Dataset*

For **Normal** and **Fire**:

Both categories were clean and free of significant issues.

For **Accidents** and **Damaged Buildings**:

**Challenges:**

- Not enough data available.
- Available datasets were very small.
- Other datasets were video frames and didn't have scene varieties.

**How we solved them:**

- Extracted the good images from the available datasets.
- Resized the images to dimensions of 224x224 pixels as a preparatory step for utilization with pretrained models.
- Applied data augmentation on the extracted images to increase the size of the dataset.
- Combined all the found images and the augmented ones into one dataset.

## 4.2 Preprocessing

### 4.2.1 Text Preprocessing

1. **Text Cleaning**
   We created a preprocessing pipeline to remove any irrelevant data from our dataset by removing numbers, symbols, and any symbols related specially to our dataset, like mentions and retweets.

2. **Remove abbreviations**
   Replaced any abbreviated words present with their elaborated versions to try and capture the full meaning behind the text.

3. **Splitting Data**
   The datasets for our project have been divided into training, testing, and validation sets according to the following distribution:
   - Training Set: 60%
   - Testing Set: 20%
   - Validation Set: 20%

4. **DistilBERT Preprocessor**
   Used DistilBERT preprocessor with the preset data that contains a tokenizer and pads the text.

### 4.2.2 Image Preprocessing

#### *4.2.2.1 Binary Image Preprocessing:*

1. **Splitting Data**
   The datasets for our project have been divided into training, testing, and validation sets according to the following distribution:
   - Training Set: 60%
   - Testing Set: 20%
   - Validation Set: 20%

2. **Keras Internal module**
   we used keras internal module to generate our images form our dataset directory.
   The main benefit of using this approach is that it performs basic data augmentation like rotation, flipping, etc., which helps in improving model generalization, also it can work with the RGB Images.

3. **Resizing**
   We resized our images to 224 x 224.

#### *4.2.2.2 Multiclass Image Preprocessing:*

1. **Splitting Data**
   The datasets for our project have been divided into training, testing, and validation sets according to the following distribution:
   - Training Set: 70%
   - Testing Set: 20%
   - Validation Set: 10%

2. **keras ImageDataGenerator**
   We used keras ImageDataGenerator to generate our images while using efficientnet_v2 preprocessor as the preprocessing function.

3. **efficientnet_v2**
   The main benefit of using efficientnet_v2 preprocessing unit performs standard preprocessing steps to make sure that all the input images are the same such as Normalization to the data and it can also work with the RGB Images.

4. **Resizing**
   We resized our images to 224 x 224.

## 4.3 Models

### 4.2.1 Text Model

**DistilBERT Transformer**
Most efficient version of BERT to work with real-time data.

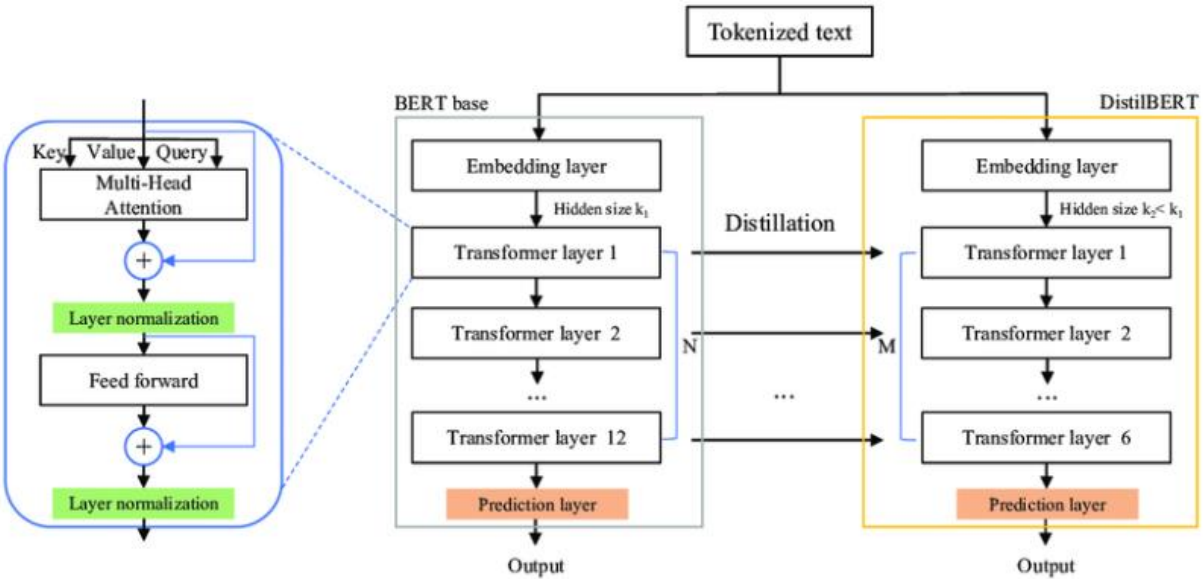


*Figure 6: DistilBERT Transformer [1]*

### 4.2.2 Image Models

#### *4.2.2.1 Binary Image Model:*

**MobileNetV3**
This model seamlessly integrated into our system, delivering exceptional speed and efficiency**.**

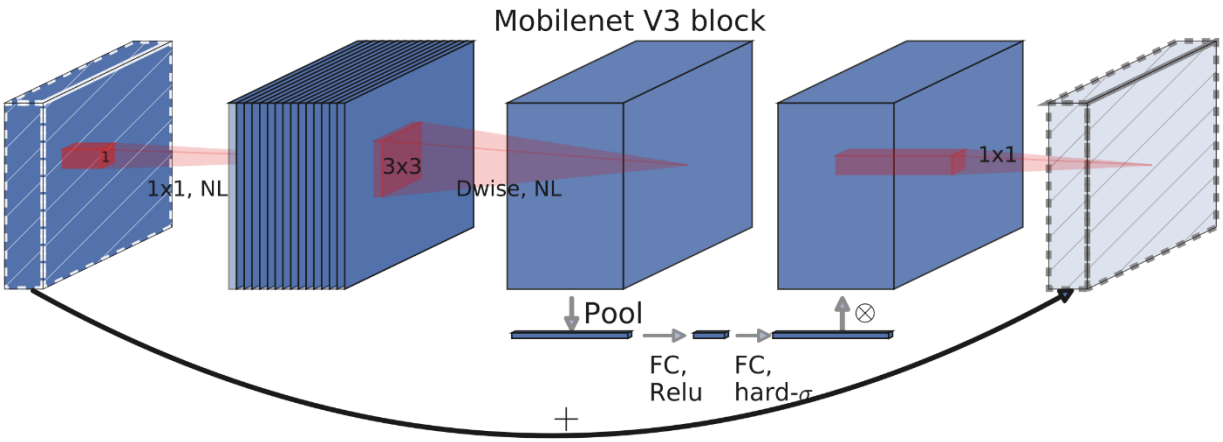


*Figure 7: Mobilenet V3[2]*

#### *4.2.2.2 Multiclass Image Model:*

**EfficientNetV2**

The most lightweight model that ensures expedited and effective performance.

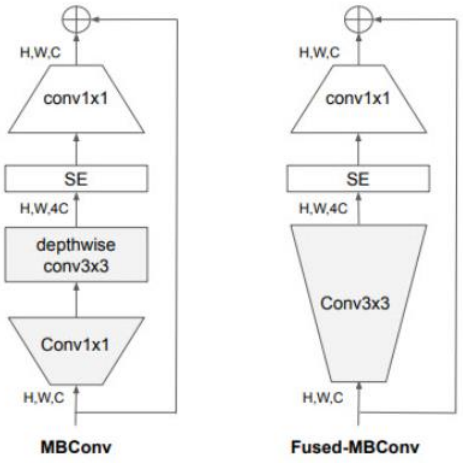


*Figure 8: EfficientNetV2[3]*

## 4.4 Experiments and Results

In this section, we explored various model approaches by trying traditional, modern, and deep learning techniques.
We enhanced each model with suitable preprocessing methods and improved their performance through hyperparameter tuning.
Finally, we documented each experiment thoroughly.

### 4.4.1 Text Results

| Model | Number of epochs | Train Accuracy | Train Loss | Validation Accuracy | Validation Loss | Test Accuracy | Test Loss |
|---|---|---|---|---|---|---|---|
| **Roberta** | **3** | **97.9%** | **0.055** | **96.97%** | **0.1031** | **96.89%** | **0.1022** |
| **DistilBERT** | **3** | **98.4%** | **0.045** | **96.78%** | **0.1061** | **96.78%** | **0.1095** |
| BERT | 3 | 98.3% | 0.047 | 96.6% | 0.1092 | 96.63% | 0.1083 |
| FNet | 4 | 98.7% | 0.037 | 96.46% | 0.11 | 96.45% | 0.11 |

*Table 7: Text Classification Results*

DistilBERT Results on our dataset:

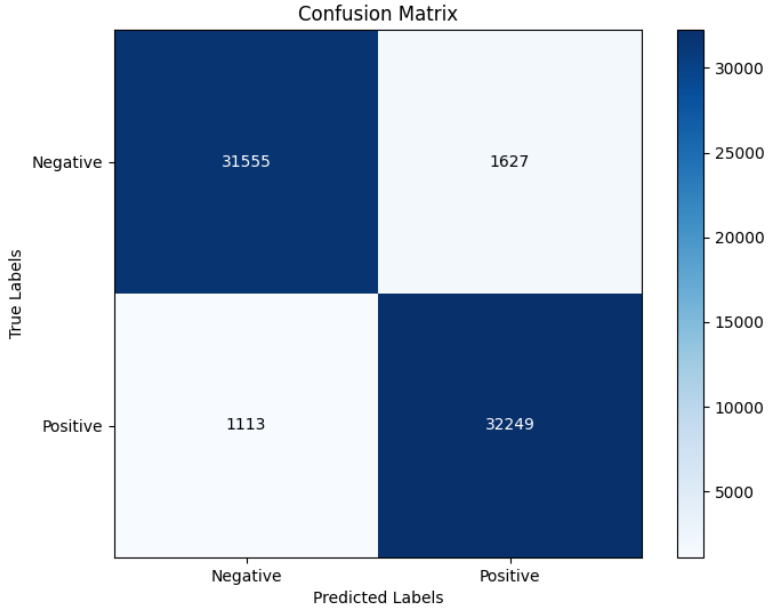


*Figure 9: DistilBERT Confusion Matrix*

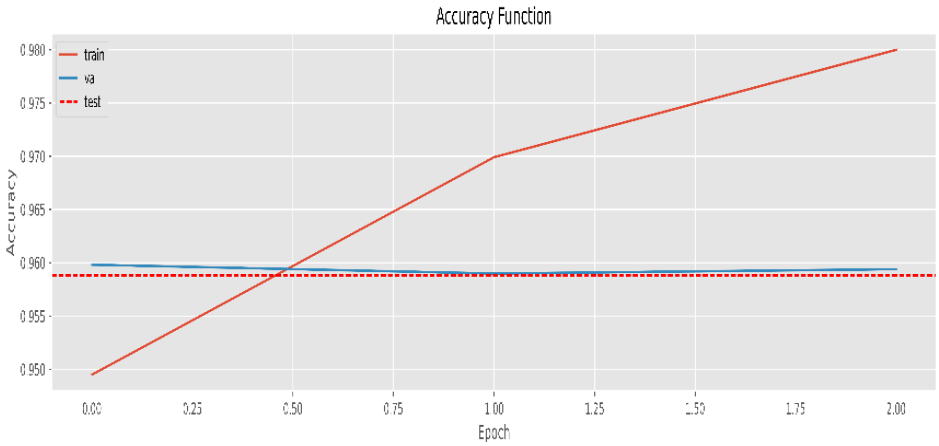


*Figure 10: DistilBERT Accuracy Function*

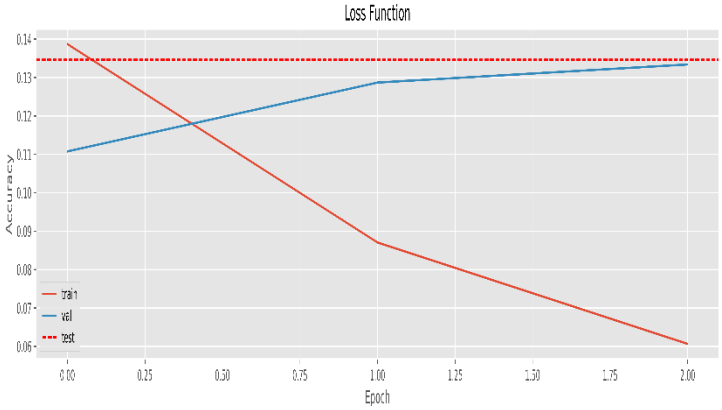


*Figure 11: DistilBERT Loss Function*

### 4.4.2 Image Results

#### *4.4.2.1 Binary Image Results:*

| Model | Number of epochs | Train Accuracy | Train Loss | Validation Accuracy | Validation Loss | Test Accuracy | Test Loss |
|---|---|---|---|---|---|---|---|
| **ResNet50** | **40** | **99.48%** | **0.0284** | **98.8%** | **0.0504** | **98.8%** | **0.0503** |
| **MobileNetV3** | **10** | **98.78%** | **0.2812** | **98.87%** | **0.9846** | **98.55%** | **0.1699** |
| EfficientNetB0 | 40 | 97.6% | 0.0671 | 97.34% | 0.0702 | 97.3% | 0.0702 |
| VGG19 | 27 | 94.7% | 0.1451 | 94% | 0.1491 | 94.6% | 0.1413 |
| DensNet121 | 40 | 91.6% | 0.2140 | 90% | 0.2664 | 91.9% | 0.2165 |
| MobileNetV2 | 40 | 89.48% | 0.2528 | 89.75% | 0.2570 | 89.74% | 0.2569 |
| Xception | 27 | 85.15% | 0.3617 | 78.8% | 0.4223 | 84.4% | 0.3422 |
| AlexNet | 20 | 80% | 0.5149 | 79.7% | 0.6003 | 79.7% | 0.6002 |
| NASNetLarge | 40 | 87.7% | 0.2935 | 78.9% | 0.4127 | 78.9% | 0.4126 |
| InceptionV3 | 9 | 75.45% | 0.5812 | 65.13% | 0.8313 | 76% | 0.4843 |

*Table 8: Binary Image Classification Results*

MobileNetV3 Results on our dataset:

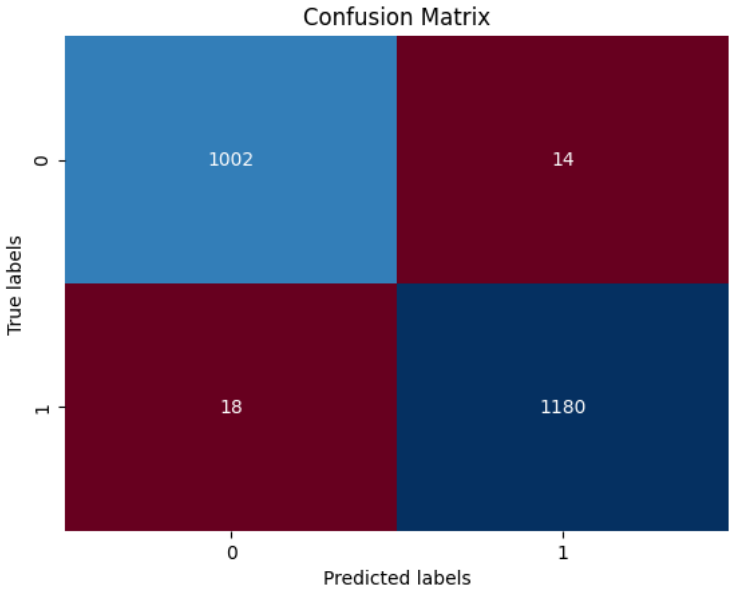


*Figure 12: MobileNetV3 Confusion Matrix*

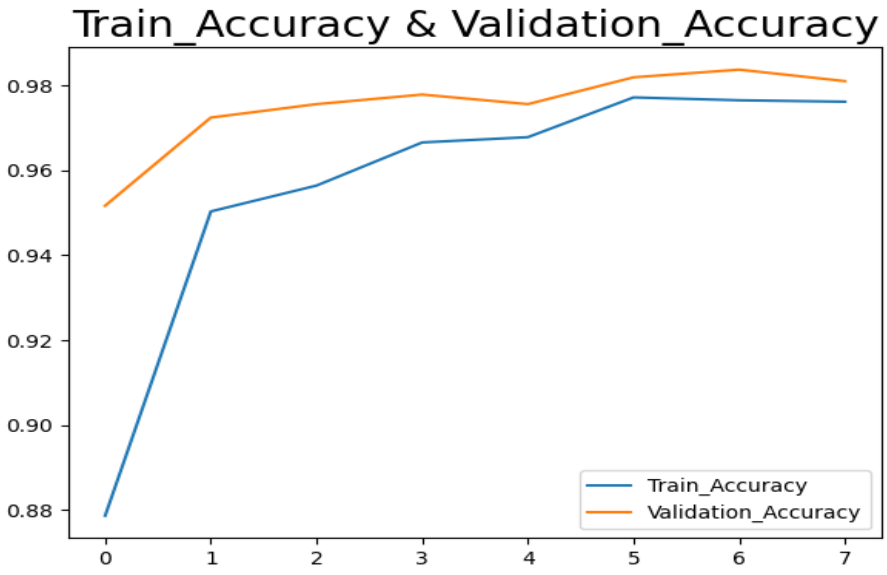


*Figure 13: MobileNetV3 Accuracy Function*

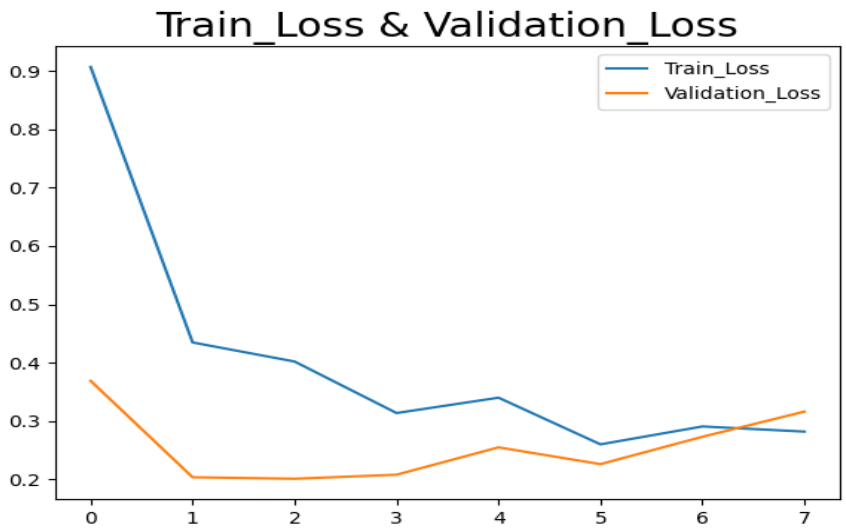


*Figure 14: MobileNetV3 Loss Function*

### *4.4.2.2 Multiclass Image Results:*

| Model | Number of epochs | Train Acc. | Train Loss | Validation Acc. | Validation Loss | Test Acc. | Test Loss |
|---|---|---|---|---|---|---|---|
| **ConvNeXt-XLarge** | **6** | **99%** | **0.0184** | **98.69%** | **0.0403** | **98.93%** | **0.0364** |
| ConvNeXtBase | 11 | 99.7% | 0.0136 | 97.9% | 0.683 | 98.1% | 0.0630 |
| ResNet50 | 13 | 99.6% | 0.0187 | 96.9% | 0.1004 | 96.95% | 0.0992 |
| **EfficientNetV2B2** | **19** | **98%** | **0.0531** | **96.7%** | **0.0968** | **96.85%** | **0.0935** |
| DenseNet201 | 10 | 99.4% | 0.0174 | 96.9% | 0.1231 | 96.7% | 0.1118 |
| MobileNetV3Large | 6 | 99% | 0.0273 | 96% | 0.1271 | 95.9% | 0.1098 |
| Xception | 25 | 97.69% | 0.1305 | 94.2% | 0.1859 | 94.3% | 0.2842 |
| NASNetMobile | 8 | 97.57% | 0.0674 | 93.79% | 0.2025 | 93.8% | 0.1804 |
| Inception-ResNet-v2 | 8 | 96% | 0.0674 | 93.5% | 0.2025 | 93.58% | 0.1926 |
| Inception | 11 | 97.65% | 0.0626 | 93.17% | 0.2308 | 93.26% | 0.211 |
| VGG16 | 25 | 69.35% | 1302.2 | 75% | 2.5057 | 74.96% | 299.9 |

*Table 9: Multiclass Image Classification Results*

EfficientNetV2B2 Results on our dataset:

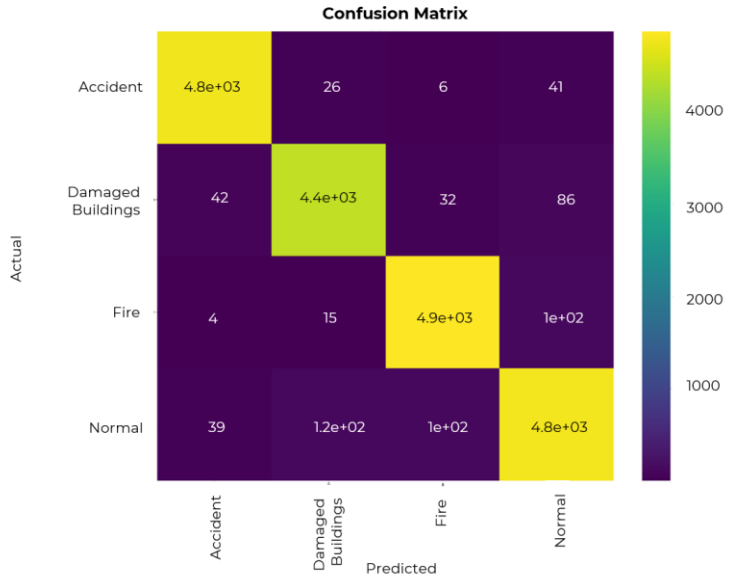


*Figure 15: EfficientNetV2B2 Confusion Matrix*

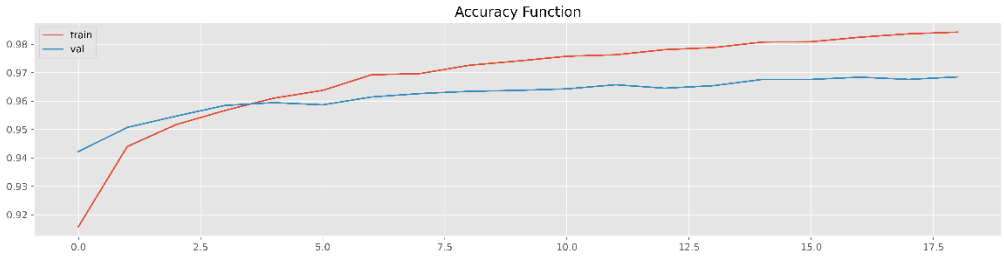


*Figure 16: EfficientNetV2B2 Accuracy Function*

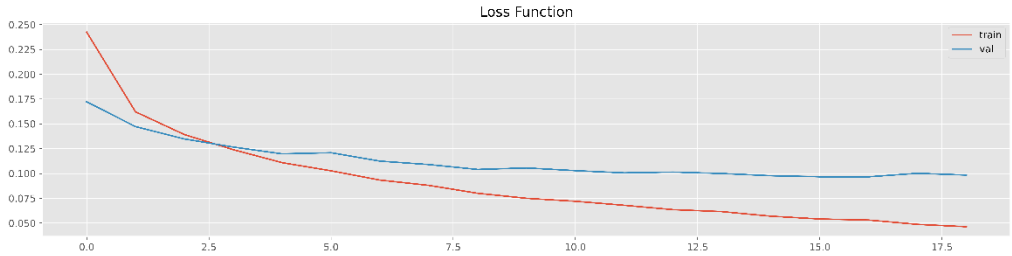


*Figure 17: EfficientNetV2B2 Loss Function*

We selected these models based on their efficiency and compact size, considering the trade-off among accuracy, performance, and efficiency within the constraints of our available resources.

## 4.5 Tools and Technologies

**Kaggle**

A machine learning and data science community that provides code, data, notebooks, courses, and competitions for data enthusiasts which are used in training the models.

The main reason why Kaggle was used is that the model development environment is that it provided the resources needed to process a huge amount of data especially the GPU resources it offered as our project's data consisted of images which was difficult to do on standard PC GPUs.

**Python**

A programming language that lets you work quickly and integrate systems more effectively.

**Keras**

A high-level neural networks API that runs on top of TensorFlow (or other backend engines).

It provides an easy-to-use interface for building and training neural networks.

**TensorFlow**

An open-source machine learning framework developed by Google.

It provides a comprehensive ecosystem for building and deploying machine learning models.

TensorFlow offers a flexible architecture with tools and libraries that allow for easy construction, training, and deployment of neural networks and other machine learning algorithms.

**HTML, CSS, jQuery, Bootstrap and JavaScript**

The core technologies for building web pages.

HTML (Hypertext Markup Language) provides the structure and content of the web page, CSS (Cascading Style Sheets) controls the presentation and styling, and JavaScript adds interactivity and dynamic behavior, jQuery and Bootstrap were used to create a responsive and interactive design.

Together, these technologies enable the creation of visually appealing, responsive, and interactive websites.

**Flask**

A popular web framework in Python used for building web applications. It provides a simple and flexible way to handle HTTP requests, define routes, and render templates.

**Git**

A distributed version control system used for tracking changes in source code during software development.
It allows multiple developers to collaborate on a project, manage different versions of the code, and facilitate code merging.

**NumPy**

NumPy is a fundamental package for scientific comp uting in Python. It provides support for large, multi-dimensional arrays and matrices, along with a wide range of mathematical functions to operate on these arrays efficiently.

**PIL**

PIL is the go-to library in Python for opening, manipulating, and saving various image file formats (JPEG, PNG, GIF, etc.). It provides a wide range of tools for tasks like resizing, cropping, rotating, color adjustments, and adding text or graphics to images.

**Requests**

The requests library simplifies making HTTP requests (GET, POST, etc.) to fetch data from web servers.

## 4.6 Functions

1. Personalized Filtering:
    - Users can fine-tune their protection through the extension settings.
    - Options include:
        - Filtering images, text, or both.
        - Creating a whitelist of images to bypass filtering.

2. Data Collection and Transmission:
    - The extension works seamlessly in the background, continuously scanning and extracting content (images and text) from the active webpage.
    - This data is transmitted to a remote server
    - The extraction process is designed to be lightweight and efficient, minimizing any impact on the user's browsing experience.

3. AI-Powered Content Analysis:
    - The server houses pre-trained deep-learning models specialized in image and text classification.
    - These models evaluate the fetched content, identifying potentially sensitive or inappropriate material.
4. Real-time Labeling & Feedback:
    - The server processes the data, generating labels that indicate the content's safety level.
    - These labels are promptly sent back to the Chrome extension.
5. Dynamic Content Blurring:
    - Based on the received labels, the extension dynamically injects a script into the webpage.
    - This script strategically blurs any content (images or text) flagged as potentially harmful, ensuring a safer browsing experience.

## 4.7 UI Design

Main Screen

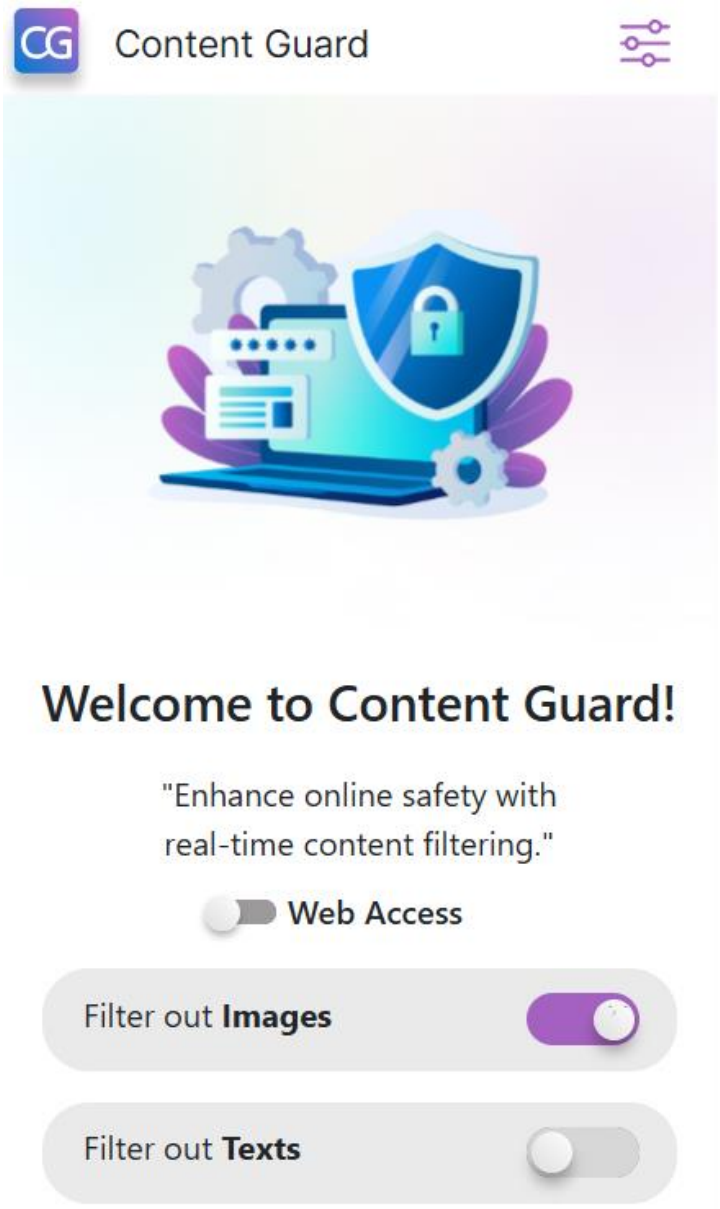


*Figure 18: Main Screen UI*

Side Menu

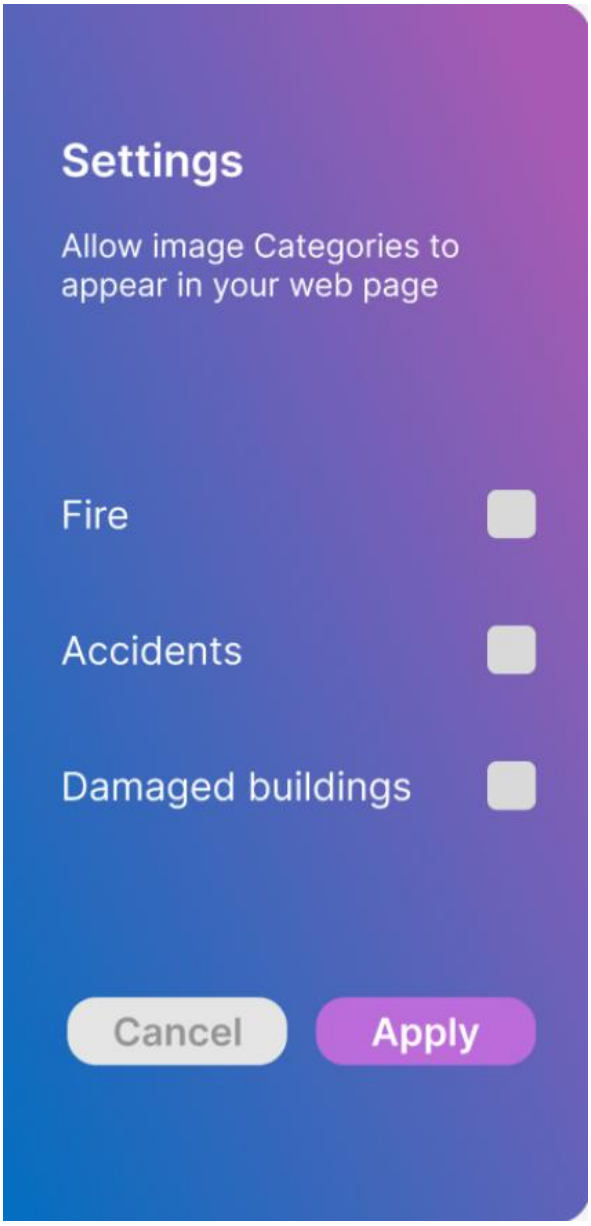


*Figure 19: Side Menu UI*

## 4.8 Wireframes

Main screen

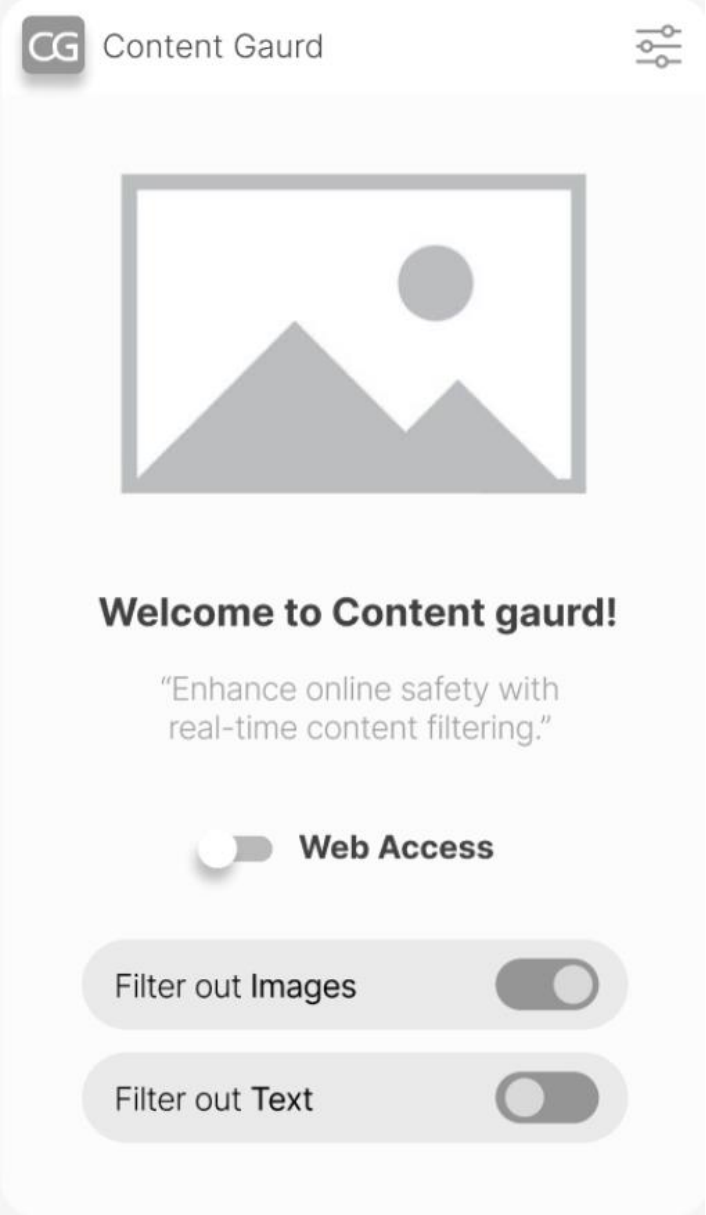


*Figure 20: Main Screen Wireframe*

Side Menu

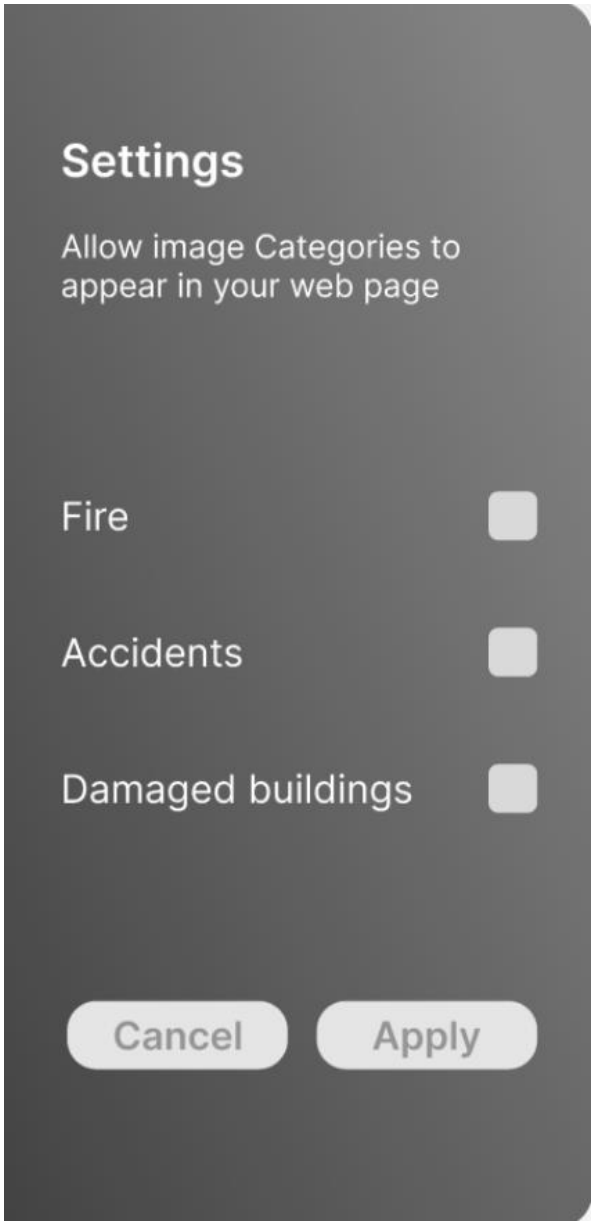


*Figure 21: Side Menu Wireframe*

# 5- User Manual

## 5.1 Installation Guide

Follow these simple steps to load the extension into your Chrome or Edge browser:

1. **Clone the Repository:**

```
git clone https://github.com/Andrew-A-A/Contient_Filter_chrome_Extenstion.git
```

2. **Open Chrome or Edge and Navigate to Extensions:**
   - For Chrome:
     - Type `chrome://extensions/` in the address bar and press Enter.
     - Alternatively, click on the three-dot menu icon in the top-right corner of Chrome, then go to `More tools` → `Extensions`.
   - For Edge:
     - Type `edge://extensions/` in the address bar and press Enter.
     - Alternatively, click on the three-dot menu icon in the top-right corner of Edge, then go to `Extensions`.

3. **Enable Developer Mode:**
   Look for the toggle switch labeled "`Developer mode`" in the top-right corner of the Extensions page and ensure it's turned on.

4. **Load the Extension:**
   For both Chrome and Edge:
   - Click on the "`Load unpacked`" button that appears after enabling Developer mode.
   - Navigate to the directory where you cloned the repository, select the folder containing the extension files, and click "`Select Folder`" or "`Open`".

5. **Confirm Loading:**
   You should now see the extension added to your list of installed extensions. Make sure it's enabled by toggling the switch if necessary.

6. **Enjoy Content Filter:**
   Content Filter is now successfully loaded into your Chrome or Edge browser!

## 5.2 User Guide

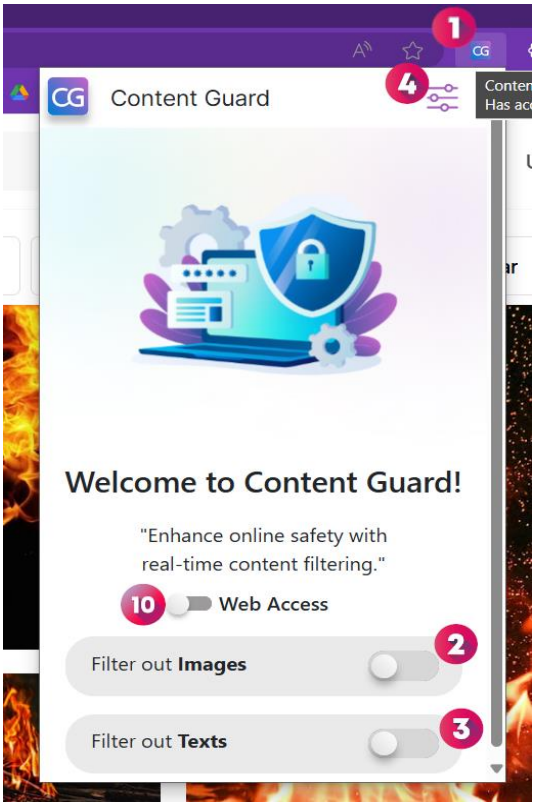


*Figure 22: Main Screen Guide*

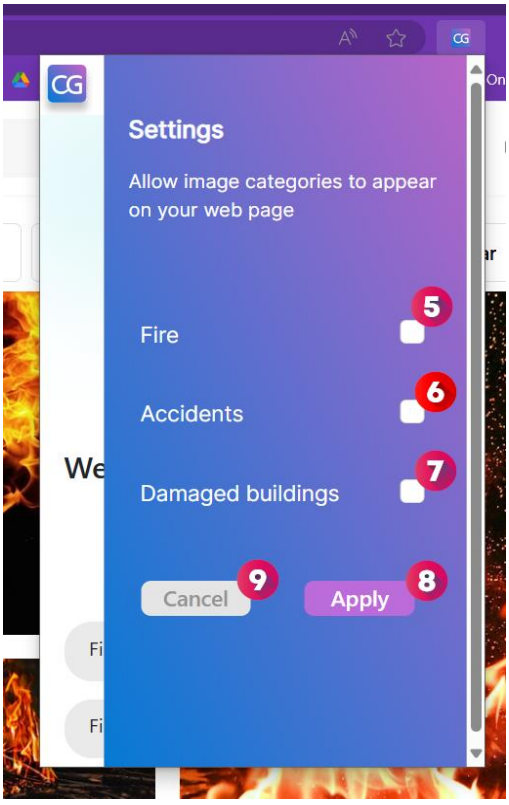


*Figure 21: Side Menu Guide*

1. Click on the Content Guard icon to open the main extension interface.

**Content Filtering Options**

2. Filter out Images: Toggle this option to filter out images.
3. Filter out Texts: Toggle this option to filter out text.

**Customization (Settings) “According to user preference”**
4. Click the Settings icon in the top right corner of the interface.

**Image Categories:**
Select one or more categories you want to add to whitelist:
5. Fire
6. Damaged buildings
7. Accidents

8. Click "Apply" to save your settings.
9. Click” Cancel” to clear your whitelist.

10. Enable "Web Access" to allow the extension to analyze web content.

# 6- Conclusion and Future Work

## 6.1 Conclusion

### Goal

Create a safer environment for our vulnerable users that do not wish to be exposed to this type of inappropriate and violent content.

### Text Classification

Performed sequence classification instead of token classification to be able to understand hidden meanings form inappropriate text.

### Image Classification

We've implemented two methods: binary and multiclass image classification. Our filtration combines the parallel predictions of both models to determine the outcome.

### Achievement

Our extension operates universally across all web pages, enhancing the browsing experience by filtering out inappropriate content from online sources, not just limited to social media platforms.

## 6.2 Future Work

### Improve performance

Enhance the accuracy of models while simultaneously reducing the latency of their predictions.

### Deploying the models

Deploy the models in the cloud to leverage the benefits of cloud computing infrastructure and services.

### Add Multilingual Feature

Enable the system to support multiple languages, expanding accessibility and enhancing user experience for a global audience.

### Text Recognition from Images

Integrate the ability to scan and read text from images, enhancing functionality by detecting whether the text should be filtered.

# References


[1] Adel, Hadeer, Abdelghani Dahou, Alhassan Mabrouk, Mohamed Abd Elaziz, Mohammed Kayed, Ibrahim Mahmoud El-Henawy, Samah Alshathri, and Abdelmgeid Amin Ali. "Improving crisis events detection using distilbert with hunger games search algorithm." Mathematics 10, no. 3 (2022): 447.

[2] Koonce, Brett, and Brett Koonce. "MobileNetV3." Convolutional Neural Networks with Swift for Tensorflow: Image Recognition and Dataset Categorization (2021): 125-144.

[3] c

[4] Sahana, V., Anil Kumar KM, and Abdulbasit A. Darem. "A Comparative Analysis of Machine Learning Techniques for Cyberbullying Detection on FormSpring in Textual Modality. “

[5] Chen, Menghuan, Jiarong Liang, Jingyao Shu, and Ren Zhu. "A Hierarchical Accident Recognition Method for Highway Traffic Systems." In Journal of Physics: Conference Series, vol. 2456, no. 1, p. 012034. IOP Publishing, 2023.

[6] Khan, Muhammad Attique, Usman Tariq, Taerang Kim, and Jae-Hyuk Cha. "Anomalous Situations Recognition in Surveillance Images Using Deep Learning.“

[7] Pericherla, Subbaraju, and E. Ilavarasan. "Overcoming the Challenge of Cyberbullying Detection in Images: A Deep Learning Approach with Image Captioning and OCR Integration." International Journal of Computing and Digital Systems 14, no. 1 (2023): 1-xx.

[8] Vijeikis, Romas, Vidas Raudonis, and Gintaras Dervinis. "Efficient violence detection in surveillance." Sensors 22, no. 6 (2022): 2216.

[9] Rubio, Jorge Luis Saavedra, Alejandro Valbuena Almeida, and Isabel Segura-Bedmar. "UC3M at Da-Vincis-2023: using BETO for Detection of Aggressive and Violent Incidents on Social Networks." In Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2023), CEUR Workshop Proceedings. CEUR-WS. org. 2023.

[10] Vallejo-Aldana, Daniel, Adrián Pastor López-Monroy, and Esaú Villatoro-Tello. "Enhancing Multi-modal Classification of Violent Events using Image Captioning." In *Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2023), CEUR Workshop Proceedings. CEUR-WS. org*. 2023.

[11] Khan, Muhammad Attique, Usman Tariq, Taerang Kim, and Jae-Hyuk Cha. "Anomalous Situations Recognition in Surveillance Images Using Deep Learning.“

[12] Ahmed, Md Tofael, Nahida Akter, Maqsuder Rahman, Dipankar Das, Touhidul AZM, and Golam Rashed. "" MULTIMODAL CYBERBULLYING MEME DETECTION FROM SOCIAL MEDIA USING DEEP LEARNING APPROACH." *International Journal of Computer Science and Information Technology (IJCSIT)* 15 (2023): 27-37.

[13] Ponce-León, Esteban, and Irvin Hussein López-Nava. "CICESE at DA-VINCIS 2023: Violent Events Detection in Twitter using Data Augmentation Techniques." In *Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2023), CEUR Workshop Proceedings. CEUR-WS. org*. 2023.

[14] Escalante-Hernandez, Alejandro, Luis Joaquín-Arellano, Jose de Jesús Lavalle-Martínez, Luis Villaseñor-Pineda, and Hugo Jair Escalante. "Towards the Monitoring of Violent Events in Social Media through Visual Information." *Computación y Sistemas* 27, no. 1 (2023): 153-162.

[15] Choqueluque-Roman, David, and Guillermo Camara-Chavez. "Weakly supervised violence detection in surveillance video." *Sensors* 22, no. 12 (2022): 4502.

[16] Bharadwaj, Vedadri Yoganand, Vasamsetti Likhitha, Vootnoori Vardhini, Adari Uma Sree Asritha, Saurabh Dhyani, and M. Lakshmi Kanth. "Automated Cyberbullying Activity Detection using Machine Learning Algorithm." In *E3S Web of Conferences*, vol. 430, p. 01039. EDP Sciences, 2023.

[17] Alduailaj, Alanoud Mohammed, and Aymen Belghith. "Detecting arabic cyberbullying tweets using machine learning." *Machine Learning and Knowledge Extraction* 5, no. 1 (2023): 29-42.

[18] Pichel, Rafael, Sandra Feijóo, Manuel Isorna, Jesús Varela, and Antonio Rial. "Analysis of the relationship between school bullying, cyberbullying, and substance use." *Children and youth services review* 134 (2022): 106369.

[19] Akhter, Arnisha, Uzzal Kumar Acharjee, Md Alamin Talukder, Md Manowarul Islam, and Md Ashraf Uddin. "A robust hybrid machine learning model for Bengali cyber bullying detection in social media." Natural Language Processing Journal 4 (2023): 100027.

[20] Singh, Vivek K., Qianjia Huang, and Pradeep K. Atrey. "Cyberbullying detection using probabilistic socio-textual information fusion." In 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 884-887. IEEE, 2016.

[21] Instagram Community Guidelines FAQ's.

[22] Facebook Community Standards | Transparency Center.

[23] X's policy on hateful conduct | X Help.

[24] Alganzory. (2024). HaramBlur: Browser extension that allows you to navigate the web with respect for your Islamic values [Software]. GitHub. https://github.com/alganzory/HaramBlur

[25] KARANDEEP, S. (2021, November). Real Life Violence and Non-Violence Data, Version 1. Retrieved December 20, 2023 from https://www.kaggle.com/datasets/karandeep98/real-life-violence-and-nonviolence-data.

[26] Abdul, M. (2023, November). Violence vs. Non-Violence: 11K Images Dataset, Version 1. Retrieved December 20, 2023 from https://www.kaggle.com/datasets/abdulmananraja/real-life-violence-situations.

[27] Monica, S. (2024, February). Text Datasets, Version 1. Retrieved December 20, 2023 from https://www.kaggle.com/datasets/monicasamehsalib/text-datasets.

[28] SHIKAMARU. (2023, September). Toxic Comments Detection, Version 1. Retrieved December 24, 2023 from https://www.kaggle.com/datasets/shikamaru073/toxic-comments-detection.

[29] CHANGLU, G. (2023, December). AIDERdata. Retrieved January 25, 2024, from https://www.kaggle.com/datasets/clguo1/aiderdata.

[30] KUTAY, K. (2021, July). Forest Fire. Retrieved January 25, 2024, from https://www.kaggle.com/datasets/kutaykutlu/forest-fire.

[31] Almond, N. (2023, December). Ukraine Images 2023. Retrieved January 26, 2024, from https://www.kaggle.com/datasets/almondnguyen/ukraine-images-2023.

[32] Arash, F. (2022, June). Wildfire surveillance. Retrieved February 10, 2024, from https://www.kaggle.com/datasets/arashfarahdel/wildfire-surveillance.

[33] Rupak, R. (2022, March). Cyclone Wildfire Flood Earthquake Database. Retrieved February 10, 2024, from https://www.kaggle.com/datasets/rupakroy/cyclone-wildfire-flood-earthquake-database.

[34] Cristian, C. (2021, July). FOREST FIRE IMAGE DATASET. Retrieved February 10, 2024, from https://www.kaggle.com/datasets/cristiancristancho/forest-fire-image-dataset.

[35] Amerzish, M. (2023, December). Forest Fire Smoke and Non-Fire Image Dataset. Retrieved February 20, 2024, from https://www.kaggle.com/datasets/amerzishminha/forest-fire-smoke-and-non-fire-image-dataset.

[36] ISMAIL, M (2021, March). The wildfire dataset. Retrieved February 20, 2024, from https://www.kaggle.com/datasets/elmadafri/the-wildfire-dataset.

[37] JOÃO, G (2021, March). Fire Images Database. Retrieved February 20, 2024, from https://www.kaggle.com/datasets/gondimjoaom/fire-images-database.

[38] Yaroslav, C (2023, Feb). RescueNet. Retrieved February 20, 2024, from https://www.kaggle.com/datasets/yaroslavchyrko/rescuenet