AN EVOLVED MODEL FOR ONLINE CONTENT FILTERING WITH REAL-TIME AI IDENTIFICATION AND IMAGERY RECOGNITION

Chenghao Feng [1], Andrew Park [2]

[1] Troy High School, 2200 Dorothy Ln, Fullerton, CA 92831
[2] California State Polytechnic University, Pomona, CA, 91768,
Irvine, CA 92620

ABSTRACT

Nexio Shield represents a significant advancement in online content moderation, leveraging AI to provide real-time protection against harmful material[1]. Our experiments demonstrate its effectiveness in detecting inappropriate content and highlight its user-friendly design. Comparative analysis with traditional moderation methods underscores its superiority in delivering immediate, unbiased, and consistent content analysis. While challenges such as reducing false negatives and enhancing customization features exist, ongoing improvements and user collaboration can enhance its effectiveness [2]. Overall, Nexio Shield contributes to creating safer online environments, addressing the limitations of traditional moderation approaches, and setting a new standard in digital safety.

KEYWORDS

Internet Security, Content Moderation, AI Detection, Image Recognition

1. INTRODUCTION

In today's digital era, the internet serves as a vast information hub and a primary means of communication. However, this openness also exposes users—especially vulnerable groups like children and teenagers—to harmful and inappropriate content. Studies show that exposure to such material can result in various negative consequences, including psychological distress and the normalization of harmful behaviors.

Traditional moderation techniques, such as manual monitoring and simple filtering systems, struggle to keep up with the immense volume of online data and the constantly evolving nature of harmful content. As a result, there is a growing need for advanced solutions that offer real-time, comprehensive protection.

In our analysis of content moderation methodologies, we compared Nexio Shield's AI-driven real-time detection approach with traditional methods: manual post-moderation, reactive moderation, and distributed moderation [3]. Manual post-moderation, involving content review after publication, often results in delayed responses to harmful material. Reactive moderation depends on user reports, leading to inconsistent and untimely interventions. Distributed moderation, where community members vote on content appropriateness, can suffer from bias

David C. Wyld et al. (Eds): NLCAI, AIFU, CCSEA, BIoT, SEA, SIPRO, BDML, CLOUD – 2025
pp. 125-135, 2025. CS & IT - CSCP 2025 DOI: 10.5121/csit.2025.151010

and lack of standardization. Nexio Shield addresses these limitations by providing immediate, unbiased content analysis through advanced AI algorithms, ensuring consistent enforcement of guidelines and enhancing user safety [4]. This proactive approach reduces exposure to harmful content and lessens the reliance on human moderators, offering a more efficient and effective solution for online content moderation.

To address these challenges, we propose Nexio Shield, an innovative online content moderation tool that uses real-time AI technology to detect and block harmful content. Nexio Shield includes features such as site blocking, AI-driven real-time detection, and image recognition to ensure a safer browsing experience [5].

Its intuitive interface and easy setup make it accessible to parents, schools, and institutions. By leveraging cutting-edge AI, Nexio Shield can instantly identify and filter inappropriate images and text, providing continuous protection while users browse. This proactive approach not only enhances safety but also offers detailed moderation performance statistics, helping users make informed decisions. Compared to traditional methods, Nexio Shield's AI-powered system is scalable and adaptable, effectively keeping up with the ever-changing digital landscape [6].

In our evaluation of Nexio Shield, we conducted two primary experiments to assess its effectiveness and user experience. The first experiment focused on the AI detection system's accuracy in identifying harmful text content across various contexts and languages. We utilized a diverse dataset to measure true positives, false positives, true negatives, and false negatives. The results indicated a high true positive rate, affirming the system's capability to detect inappropriate content effectively, though some false negatives highlighted areas for improvement. The second experiment assesses the accuracy of the moderation system for harmful image content. This is a domain that CNNs are particularly well suited for and thus we found great success in achieving a high degree of accuracy [7].

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Harmful Content

A key feature of Nexio Shield is its AI-driven real-time detection system. One of the primary challenges in implementing this system is ensuring that the AI model accurately identifies a wide range of harmful content across different contexts and languages [8].

To address this, we can train the model using a diverse dataset that covers various cultural and linguistic nuances. Additionally, integrating a continuous learning mechanism will allow the system to update itself as new types of harmful content emerge. A convolutional neural network architecture will be ideal given its ability to understand multi-word contexts; it also works as a better alternative to transformers for our use case since a global context space is not needed for our purposes.

2.2. Inappropriate Images

Another essential component is the image recognition feature. The challenge here is ensuring that the system correctly identifies inappropriate images without generating false positives that block harmless content.

To improve accuracy, we can implement advanced image analysis techniques and regularly update the recognition algorithms. Incorporating user feedback will also help refine the system over time, ensuring that it remains effective and minimizes unnecessary content restrictions.

2.3. The User Interface and User Experience

The user interface (UI) and user experience (UX) design play a crucial role in the system's success [9]. The challenge is to develop an interface that is both user-friendly and comprehensive, catering to individuals with varying levels of technical expertise.

Conducting user testing sessions will allow us to gather feedback and continuously improve the design. Ensuring a simple setup process and customizable protection settings will be critical for widespread adoption.

3. SOLUTION

Nexio Shield consists of three core components: the AI-driven real-time detection system, the image recognition module, and the user interface/dashboard.

The system operates by monitoring web content as users browse. The AI detection system analyzes text in real time, identifying and blocking harmful content [10]. Simultaneously, the image recognition module scans images to detect inappropriate visuals. These two components work together seamlessly, providing comprehensive protection.

The user interface allows users to view detailed statistics on content moderation performance, customize protection settings, and receive real-time updates on detected and filtered content. The system is designed for simplicity, ensuring that users can easily navigate and manage their online safety.

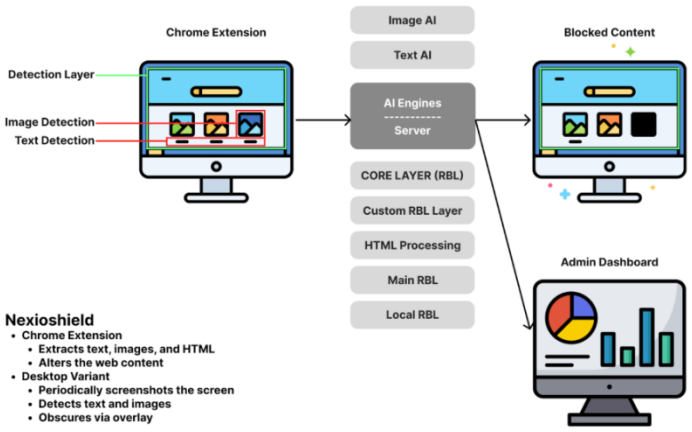


Figure 1. Overview of the solution

The AI-driven real-time detection system is designed to analyze text content during browsing sessions, immediately identifying and blocking inappropriate material. It employs Natural Language Processing (NLP) techniques combined with the capabilities of a Convolutional Neural Network to understand context and semantics on a character level, ensuring accurate detection across various scenarios [11].

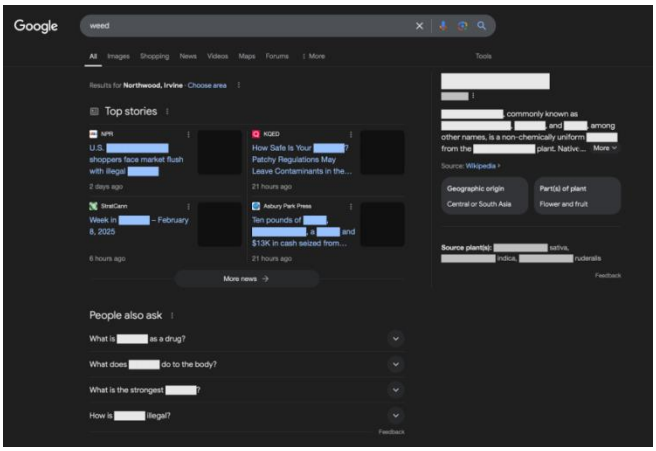


Figure 2. Screenshot of Google research

```
/*
CORE LAYER (RBL) (Always Blocked Websites/Always Allowed Websites)
Custom RBL Layer (Blocked sites requested by c_user admin)
HTML Processing (Standard allowed/unknown sites)
[Redundant] Main RBL Checking if Above Fails.
[Redundant] Local RBL Checking if Above Fails.
*/

if (request.action === "censor") {
    console.log("censoring");
    const response = request.response;
    console.log("response: ", response);
    // go through the innerHTML of the page and replace any flagged sentences
    var sentences = response.sentences;
    // separate the sentences blob into an array of sentences while also removing any special characters
    const sentencesBlob = sentences.join(" ");
    const sentencesArray = sentencesBlob.split(/(?<=[.!?])\s+/);
    sentences = sentencesArray.map((sentence) => {
      return sentence.replace(/[^a-zA-Z0-9 ]/g, "");
    });
    console.log("sentences: ", sentences);
    function replaceTextInNode(node, sentences) {
      // Iterate over each sentence and replace it in the text content of the node
      sentences.forEach((sentence) => {
        const regex = new RegExp(sentence, "gi");
        let paragraphCharCount = sentence.length;
        console.log("Replacing ", sentence, " with ", "".repeat(paragraphCharCount));
        node.nodeValue = node.nodeValue.replace(regex, "".repeat(paragraphCharCount));
      });
    }
    function traverseDOMAndReplaceText(root, sentences) {
      const walker = document.createTreeWalker(root, NodeFilter.SHOW_TEXT, null, false);
      let node;
      while ((node = walker.nextNode())) {
        replaceTextInNode(node, sentences);
      }
    }
    traverseDOMAndReplaceText(document.body, sentences);
    // replace the image links that need to be censored
    const images = request.images;
    images.forEach((image) => {
      console.log("Checking if ", image, " is in the page");
      const regex = new RegExp(image, "gi");
      newHTML = newHTML.replace(regex, chrome.runtime.getURL("assets/images/blocked.svg"));
    });
}
```

Figure 3. Screenshot of code 1

This code runs within a Chrome extension as part of a content moderation system. It executes when a `censor` action is received, triggered after the extension communicates with a backend server that screens webpage content. The backend analyzes the text on a webpage and returns flagged content. ALong the way, the text also travels through a series of RBL layers to help optimize the workload of the underlying text AI models [12]. The `response` object contains flagged sentences. First, the code extracts sentences from `response.sentences`, processes them by removing special characters, and splits them into an array. Then, it traverses the DOM using `traverseDOMAndReplaceText`, which finds text nodes and replaces flagged sentences with a corresponding number of "" characters. The `replaceTextInNode` function ensures each flagged phrase is completely censored. For images, which are flagged by a companion module, the script checks if flagged URLs appear on the page and replaces them with a "blocked" image hosted in

the extension's assets [13]. This ensures both offensive text and images are dynamically censored on the user's browser.

The image recognition module scans and analyzes images in real time to detect inappropriate visuals. It leverages deep learning and computer vision techniques using a YOLO-based model for object detection. This ensures rapid and precise filtering of harmful images, integrating seamlessly with the AI detection system for comprehensive content moderation.

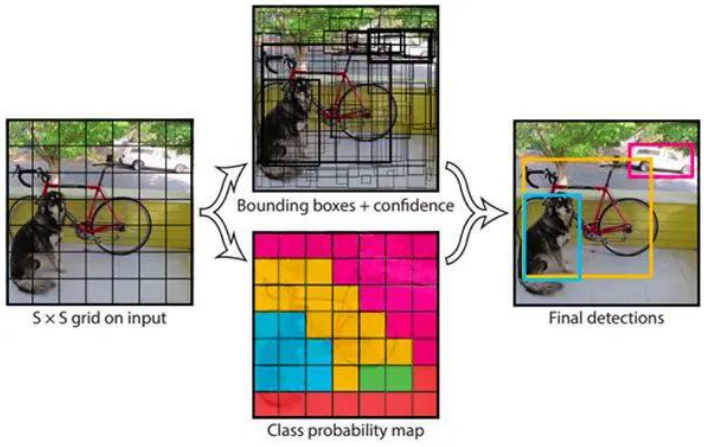


Figure 4. A simplified illustration of the YOLO object detection pipeline (source)

```
from ultralytics import YOLO

import torch

import torchvision

def main():

  print("Torch version:", torch.__version__)

  print("Torchvision version:", torchvision.__version__)

  print("CUDA available:", torch.cuda.is_available())

  print("Number of GPUs:", torch.cuda.device_count())

  print("Current GPU:", torch.cuda.current_device())

  print("GPU Name:", torch.cuda.get_device_name(torch.cuda.current_device()))
```

```
  # Verify if CUDA is available

  if torch.cuda.is_available():

      device = 'cuda'

  else:

      device = 'cpu'

      print("CUDA not available, using CPU")

  # Define paths

  data_config_path = 'data.yaml'

  model_save_path = 'model.pt'

  # Initialize and train the YOLO model

  model = YOLO("yolov8n.pt")  # Change to yolov7 or yolov9 if using those versions

  model.train(data=data_config_path, epochs=50, batch=16, imgsz=640, device=device)

  # Evaluate model

  results = model.val(data=data_config_path, device=device)

  metrics = results.results_dict  # Get the results dictionary

  accuracy = metrics['metrics/precision(B)']  # Adjust to the correct metric key if needed

  print(f'Validation Precision: {accuracy * 100:.2f}%')

  # Save the trained model

  model.save(model_save_path)


if __name__ == "__main__":

  main()
```

Figure 5. Screenshot of code 2

This script trains a YOLO (You Only Look Once) model on a dataset using the `ultralytics` library [14]. It runs on a server, likely one equipped with a GPU for accelerated training. The script first checks the system's PyTorch installation and GPU availability. If a CUDA-compatible GPU is available, it sets the training device to `'cuda'`; otherwise, it falls back to CPU. The training process uses a specified dataset configuration file (`data.yaml`) and initializes a pretrained YOLOv8 model (`yolov8n.pt`). The model is then trained for 50 epochs with a batch size of 16 and an image size of 640x640 pixels. After training, it evaluates the model on validation data, retrieving the precision metric to gauge its accuracy. Finally, the trained model is saved as `model.pt` for deployment. The backend server uses this model for object detection, likely integrating it into an API for real-time content moderation or image classification.

The user interface/dashboard provides guardians and administrators with control over the system, allowing them to view detection statistics, adjust moderation settings, and receive real-time alerts. Built with Elysia.js and MySQL, it ensures secure and intuitive access to content filtering data. The dashboard enhances user experience by making content moderation transparent and customizable.

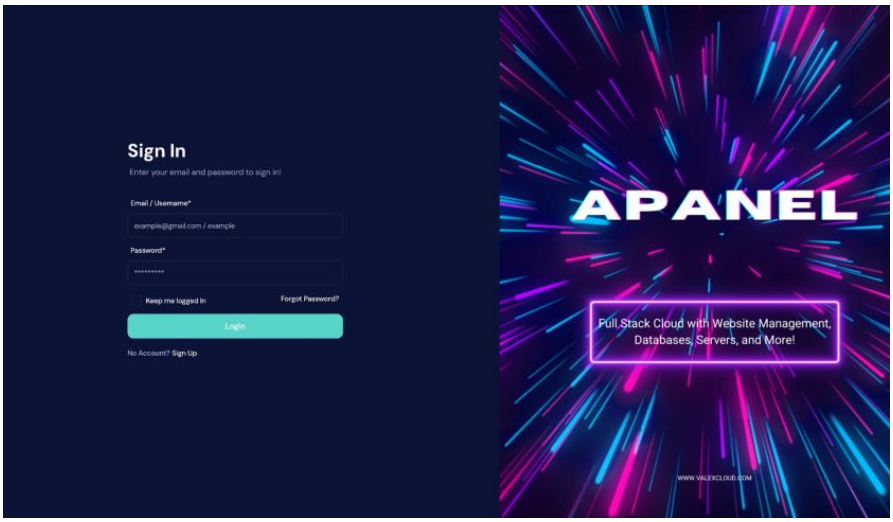
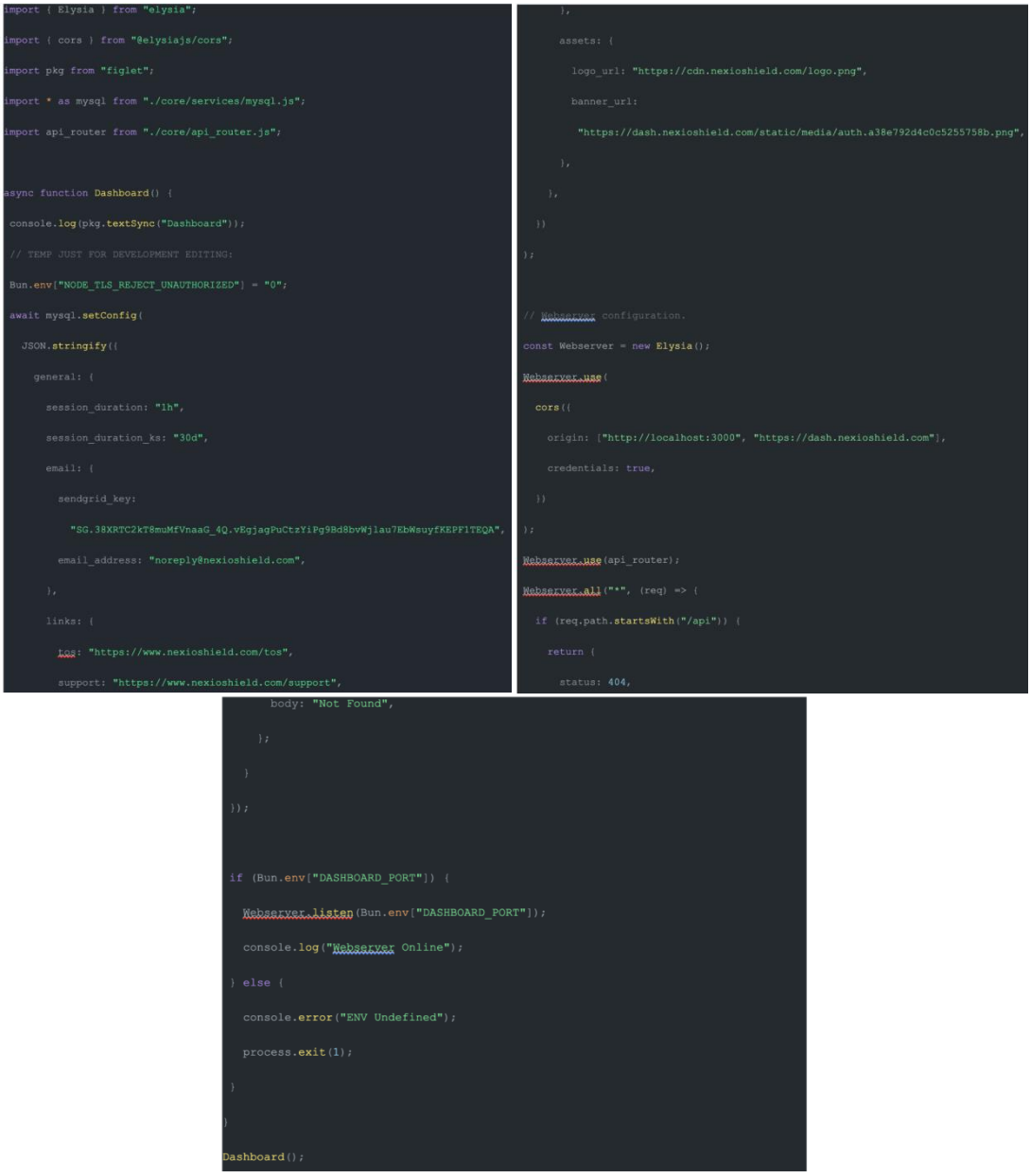


Figure 6. Screenshot of sign in page



```
import { Elysia } from "elysia";
import { cors } from "@elysiajs/cors";
import pkg from "figlet";
import * as mysql from "./core/services/mysql.js";
import api_router from "./core/api_router.js";

async function Dashboard() {
 console.log(pkg.textSync("Dashboard"));
 // TEMP JUST FOR DEVELOPMENT EDITING:
 Bun.env["NODE_TLS_REJECT_UNAUTHORIZED"] = "0";
 await mysql.setConfig(
   JSON.stringify({
     general: {
       session_duration: "1h",
       session_duration_ks: "30d",
       email: {
         sendgrid_key:
           "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",
         email_address: "noreply@nexioshield.com",
       },
       links: {
         tos: "https://www.nexioshield.com/tos",
         support: "https://www.nexioshield.com/support",
       },
       assets: {
         logo_url: "https://cdn.nexioshield.com/logo.png",
         banner_url:
           "https://dash.nexioshield.com/static/media/auth.a38e792d4c0c5255758b.png",
       },
     },
   })
 );

// Webserver configuration.
const Webserver = new Elysia();
Webserver.use(
  cors({
    origin: ["http://localhost:3000", "https://dash.nexioshield.com"],
    credentials: true,
  })
);
Webserver.use(api_router);
Webserver.all("*", (req) => {
  if (req.path.startsWith("/api")) {
    return {
      status: 404,
      body: "Not Found",
    };
  }
});

if (Bun.env["DASHBOARD_PORT"]) {
  Webserver.listen(Bun.env["DASHBOARD_PORT"]);
  console.log("Webserver Online");
} else {
  console.error("ENV Undefined");
  process.exit(1);
}
}
Dashboard();
```

Figure 7. Screenshot of code 3

This script sets up an admin dashboard using Elysia, allowing parents, guardians, and schools to monitor and control the content moderation extension. It initializes the server by printing "Dashboard" in ASCII text using `figlet` and configures the MySQL database with session durations, email settings (via SendGrid API), and links to the Terms of Service and support. The server is built with Elysia and uses CORS to allow requests from `localhost:3000` and `dash.nexioshield.com`. It integrates an API router (`api_router.js`) to handle moderation settings and user management, ensuring API routes are properly defined while returning 404 errors for undefined endpoints. The script checks if a `DASHBOARD_PORT` is set and starts the server; otherwise, it logs an error and exits. This dashboard likely provides an interface for viewing flagged content, configuring censorship rules, and monitoring user activity, making it a crucial component for managing the Chrome extension's content filtering system.

4. EXPERIMENT

4.1. Experiment 1

This experiment assesses Nexio Shield's ability to accurately identify harmful text across different languages and contexts.

To evaluate detection accuracy, we would compile a diverse dataset containing various types of words, phrases, and sentences labeled as either appropriate or, if flagged, which type of inappropriate category it falls under. The dataset would span multiple languages and cultural contexts to ensure a comprehensive assessment. Nexio Shield's AI system would analyze this dataset, and we would track how often it correctly detects harmful content (true positives), fails to detect it (false negatives), incorrectly flags safe content (false positives), and correctly identifies safe content (true negatives). This setup allows for a thorough evaluation of the AI's performance across different scenarios.

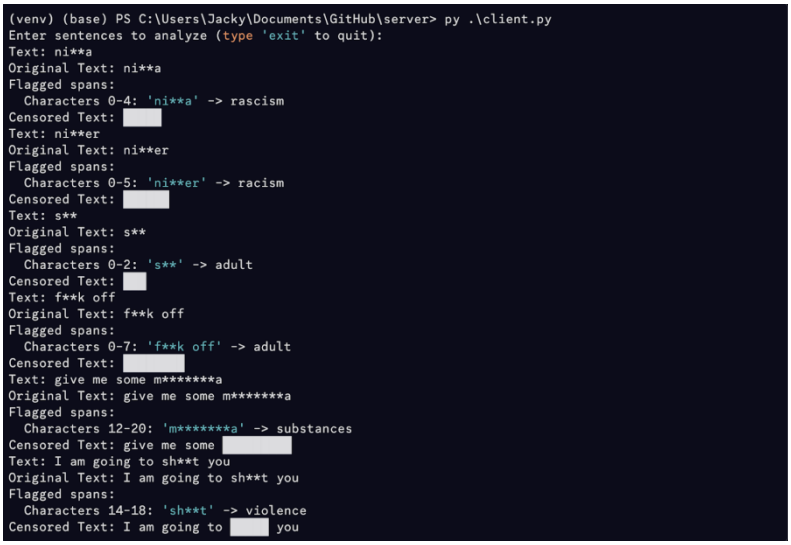


Figure 8. Actual inputs partially censored with "*"

The results of this experiment showed that Nexio Shield was generally effective in detecting explicit or harmful language across various contexts. It consistently identified slurs, adult language, references to violence, and drug-related content with appropriate flagging and censoring. However, certain edge cases revealed blind spots. For example, while direct profanity and well-known slurs were caught reliably, more ambiguous or phonetically censored terms might bypass detection, especially when spaced, stylized, or misspelled. The most surprising finding was how well the model handled complex, multi-word expressions like "f**k off" or contextually aggressive phrases like "I am going to sh**t you." This implies that contextual phrase recognition is a strength. On the other hand, potential weaknesses may emerge when facing multilingual input or slang terms outside the training data. The biggest influence on results appears to be the diversity and representativeness of the dataset—terms that were underrepresented in training were more likely to slip through undetected.

4.2. Experiment 2

The second domain we need to test for accuracy is in images, the model for which is built on top of the YOLO models for classifications.

To evaluate image detection accuracy, we can compile a variety of images matching each of the targeted categories along with benign images that do not fall into any of them. The underlying CNN model will then take in foreign images for testing. Nexio Shield's AI system would analyze this activity, and we would track how often it correctly detects harmful images (true positives), fails to detect it (false negatives), incorrectly flags safe content (false positives), and correctly identifies safe content (true negatives).

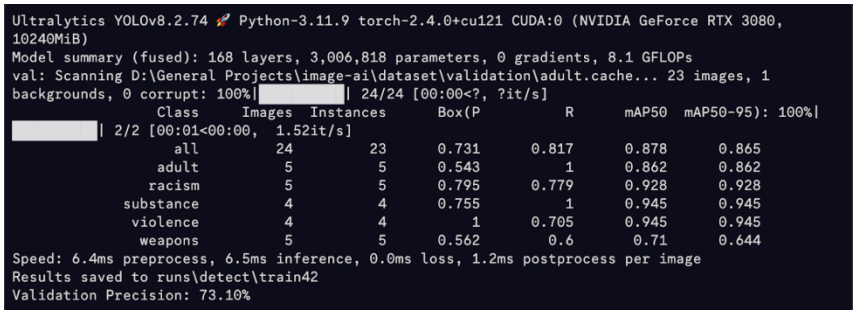


Figure 9. Figure of experiment 2

The image AI proved to be very accurate from preliminary testing, as nearly all classifications were able to achieve ideal thresholds for initial production. The only exception was the weapons category, which performed below expectations. This is likely due to the dataset not including enough representative images in terms of both quantity and diversity. Analyzing the data, the mAP50 values ranged from 0.71 (lowest, for weapons) to 0.945 (highest, for substance and violence). The mean mAP50 is approximately 0.877, while the median is 0.878, indicating consistent high performance across most classes. The mAP50–95 values follow a similar pattern, with a mean and median around 0.865 and 0.862, respectively. What stood out was the extremely high recall (R) values for some categories, like adult and substance (both at 1.0), while weapons had the lowest at 0.6. This suggests that recall is heavily influenced by dataset balance and quality, which likely has the biggest effect on overall model performance.

5. RELATED WORK

Manual post-moderation involves reviewing content after publication, with human moderators assessing user-generated material for guideline compliance [15]. While this ensures engagement by allowing immediate posting, it is labor-intensive and slow in addressing harmful content. TikTok employs AI-driven moderation alongside human oversight, but studies indicate that its AI systems often fail to remove harmful material effectively, leading to gaps in user protection (Ambran et al., 2024). Our solution improves upon this by integrating real-time filtering directly within a Chrome extension. Unlike TikTok's approach, our system operates at the user level, allowing parents and teachers to monitor and control content filtering through an admin dashboard.

Reactive moderation relies on users to report inappropriate content, which is then reviewed by moderators [16]. While platforms like YouTube rely on user reports and automated detection, research shows that harmful children's videos often bypass these filters and remain available for

extended periods (Ahmed, 2023). Our approach eliminates this delay by proactively filtering content before it reaches the user. The Chrome extension sends text and image data to a server, which evaluates and censors the content in real-time. This ensures immediate intervention, reducing users' exposure to harmful material without relying on user reports.

Distributed moderation relies on community voting to determine content appropriateness, offering scalability but leading to inconsistency and bias [17]. Research highlights that platforms applying distributed moderation often enforce guidelines inconsistently, disproportionately affecting marginalized groups (Saldías, 2024). Our solution avoids this issue by employing AI-driven filtering with centralized control through the admin dashboard. Unlike community-driven models, our system ensures uniform enforcement while granting parents and teachers the ability to adjust strictness settings, creating a balance between automated filtering and human oversight tailored to individual user needs.

6. CONCLUSIONS

While Nexio Shield offers robust real-time content moderation, it may still struggle at times with contextual interpretation, potentially leading to false positives or negatives. Cultural nuances and evolving language trends also present challenges for AI detection [18]. To mitigate these limitations, continuous updates incorporating diverse datasets are essential. Implementing a user feedback system where individuals can report misclassified content—whether harmful content that was missed or safe content that was incorrectly flagged—can further enhance accuracy. Additionally, incorporating a human-in-the-loop approach for complex moderation decisions can complement AI efficiency with human judgment, improving overall effectiveness.

Nexio Shield represents a significant advancement in online content moderation, leveraging an AI suite to provide real-time protection against harmful material. While challenges remain, ongoing improvements and user collaboration can further refine its capabilities, contributing to safer and more reliable online environments.

REFERENCES

[1] Gallego, Francisco, Ofer Malamud, and Cristian Pop-Eleches. Parental monitoring and children's Internet use: The role of information, control, and cues. No. w23982. National Bureau of Economic Research, 2017.

[2] Eales, Lauren, et al. "Children's screen and problematic media use in the United States before and during the COVID-19 pandemic." Child development 92.5 (2021): e866-e882.

[3] Fenton, Stephen J., et al. "The Utah Pediatric Trauma Network, a statewide pediatric trauma collaborative can safely help nonpediatric hospitals admit children with mild traumatic brain injury." Journal of trauma and acute care surgery 95.3 (2023): 376-382.

[4] Stoilova, Mariya, Monica Bulger, and Sonia Livingstone. "Do parental control tools fulfil family expectations for child protection? A rapid evidence review of the contexts and outcomes of use." Journal of Children and Media 18.1 (2024): 29-49.

[5] Stoev, Martin, and Dipti K. Sarmah. "Online protection for children using a developed parental monitoring tool." International Congress on Information and Communication Technology. Singapore: Springer Nature Singapore, 2023.

[6] Hernandez, J. Maya, et al. "Parental monitoring of early adolescent social technology use in the US: a mixed-method study." Journal of Child and Family Studies 33.3 (2024): 759-776..

[7] Anderson, Monica. "1. How parents monitor their teen's digital behavior." Pew Research Center: Internet, Science & Tech (2016).

[8] Roberts, Kim P., Katherine R. Wood, and Breanne E. Wylie. "Children's ability to edit their memories when learning about the environment from credible and noncredible websites." Cognitive Research: Principles and Implications 6.1 (2021): 42.

[9] Ponti, Michelle. "Screen time and preschool children: Promoting health and development in a digital world." Paediatrics& child health 28.3 (2023): 184-192.
[10] Gentile, Douglas A., et al. "Protective effects of parental monitoring of children's media use: A prospective study." JAMA pediatrics 168.5 (2014): 479-484.
[11] Behrens, Sarah, Evan Dean, and Marisol Torres. "Family perspectives on developmental monitoring: A qualitative study." Developmental Disabilities Network Journal 2.2 (2022): 8.
[12] Muppalla, Sudheer Kumar, et al. "Effects of excessive screen time on child development: an updated review and strategies for management." Cureus 15.6 (2023).
[13] Arumugam, Catherine Thamarai, Mas Ayu Said, and Nik Daliana Nik Farid. "Screen-based media and young children: Review and recommendations." Malaysian Family Physician: the Official Journal of the Academy of Family Physicians of Malaysia 16.2 (2021): 7.
[14] Margalit, Liraz. "What screen time can really do to kids' brains." Psychology Today (2016).
[15] Ambran, Nur Syafiqah, Wan Hartini Wan Zainodin, and Muhammad Naim Muhamad Ali. "AI SYSTEMS AND CONTENT MODERATION TIKTOK AS A DIGITAL SAFETY PLATFORM IN SHAPING A PLEASANT ENVIRONMENT: A QUALITATIVE APPROACH." Journal of Media and Information Warfare Vol 17.1 (2024): 93-104.
[16] Ahmed, Syed Hammad. "A Multimodal Framework for Automated Content Moderation of Children's Videos." (2024).
[17] Saldías, Belén. "Designing Child-Centered Content Exposure and Moderation." arXiv preprint arXiv:2406.08420 (2024).
[18] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.