JSRT JOURNAL
PAGES: 189-203
26/06/25
JOURNAL OF SCIENTIFIC RESEARCH AND TECHNOLOGY VOLUME-3 ISSUE-6 JUNE

On-Screen Activity Classification In E-Learning: A Federated Learning Approach For Privacy Preservation

Saniya Iram Khan[1], Dr. Shameem Akther[2]

[1]*Student, Dept.Of Computer Science and Engineering, Khaja BandaNawaz University, Kalaburagi, India.*

[2]*Assistant ProfessorDept. Of Computer Science and Engineering, Khaja BandaNawaz University, Kalaburagi, India.*

ABSTRACT

With the rapid rise of online education, students often use the same devices for both learning and entertainment. This blurred boundary between productive and non-productive activities causes distraction and reduced learning efficiency. Existing solutions such as screen monitoring software or activity tracking tools either lack accuracy or severely compromise user privacy by transmitting raw screen data to centralized servers.

This project proposes a privacy-preserving approach that tracks on-screen activities, classifies them using deep learning, and ensures data privacy via federated learning. Screenshots are captured locally every minute and classified into categories such as programming, youtube_edu, or youtube_entertainment. The system provides users with a real-time, interactive dashboard to visualize how much time is spent on each activity type. Federated learning ensures that the deep learning model is trained and updated across devices without ever sharing raw data, thus offering both accuracy and privacy.

Keywords: Screen Activity, Python, Youtube.

1. INTRODUCTION

In recent years, e-learning has witnessed exponential growth due to technological advancements and the widespread availability of internet connectivity. Digital learning platforms like Coursera, Udemy, YouTube, and Google Classroom have transformed the traditional education system into a more flexible and accessible environment. However, this transformation has introduced new challenges in maintaining student engagement and productivity during online learning sessions.

One of the primary concerns in online education is the inability to monitor and manage how students utilize their screen time. Unlike in physical classrooms, where the teacher can observe student behavior directly, online learning environments offer minimal visibility into whether students are focused on academic content or distracted by entertainment. The problem is further complicated by the fact that platforms like YouTube serve both educational and entertainment content, making it difficult to determine the user's intent based solely on the platform being accessed.

While existing screen monitoring solutions attempt to address this issue by capturing screenshots or tracking website URLs, these approaches raise significant concerns regarding privacy. Transmitting screen content or user activity logs to centralized servers can potentially expose sensitive personal information, violating user trust and ethical data handling practices.

To address these challenges, this project presents a privacy-preserving on-screen activity tracking and classification system using federated learning. The system captures screen images locally at fixed intervals and classifies the activity into predefined categories such as "programming," "youtube_edu," or "youtube_entertainment" using a deep learning model. Unlike conventional systems, this approach ensures that user data never leaves the local device. Instead, only model updates are shared during training, thereby maintaining user privacy while still enabling collaborative model improvement.

The solution also features a real-time web-based dashboard that provides insights into the user's daily screen usage patterns. This empowers students to self-regulate their study habits, improve focus, and make informed decisions about their online behavior.

By combining deep learning, federated learning, and modern web technologies, this system offers a comprehensive, secure, and user-friendly approach to improving the effectiveness of e-learning environments.

2. PROBLEM STATEMENT

With the transition from traditional classroom environments to digital learning platforms, ensuring student focus and productivity has become increasingly difficult. In most cases, students use the same digital devices for both academic purposes and leisure activities. While they may access platforms like YouTube for educational videos, the ease of transitioning to unrelated entertainment content, such as music videos or social media, poses a significant distraction.

Existing monitoring tools attempt to track digital behavior through continuous screen recording, keystroke logging, or URL monitoring. However, these tools often:

- Require invasive access to user devices,
- Violate user privacy by transmitting sensitive information to central servers,
- Cannot distinguish between productive and unproductive activities on platforms that serve both (e.g., YouTube),
- Provide little feedback to users regarding their own digital habits.

This leads to several challenges:

1. Lack of real-time, automated classification of screen content.
2. Inability to distinguish educational vs. entertainment content within the same platform.
3. Absence of privacy-preserving mechanisms in existing screen tracking systems.
4. Limited user feedback and visualization tools to promote self-awareness and improvement.

There is a need for a system that can track screen activities, intelligently classify them, and help users understand their digital usage patterns—without compromising personal privacy. The solution must enable real-time insights, avoid raw data transmission, and respect user autonomy in a non-invasive and ethical manner.

This project aims to address these challenges by designing and implementing a screen activity classification and tracking system based on federated learning principles, using lightweight deep learning models and an intuitive, interactive user dashboard.

3. LITERATURE SURVEY

The growing dependence on online education has led to extensive research into methods that can track, analyze, and improve student engagement and productivity. Several approaches have been proposed in the domains of computer vision, machine learning, and privacy-preserving technologies. This section reviews relevant literature in these areas, with a focus on activity classification, screen monitoring, and privacy.

3.1 Screen Monitoring Technologies

Traditional screen monitoring software such as **Hubstaff**, **Teramind**, **ActivTrak**, and **DeskTime** provide functionalities like screenshot capturing, URL tracking, application monitoring, and keystroke logging. While effective in workplace monitoring, these solutions raise significant concerns in educational settings due to:

- Continuous screen recording that can capture sensitive or personal content.
- Centralized storage of user data, increasing the risk of privacy violations.
- High computational overhead in capturing and transmitting video or image data.

Studies such as Hankison et al. (2021) have raised ethical concerns about surveillance software, especially when used to monitor students without explicit consent. These systems, although efficient in logging usage, do not differentiate between the nature of activities (e.g., educational vs. entertainment) within platforms like YouTube.

3.2 Machine Learning for Activity Detection

Recent studies have introduced machine learning models to classify screen activity. Ferdousi et al. (2021) proposed a deep learning-based system using ResNet50 and VGG16 to classify screenshots into different categories. While the models achieved promising accuracy (95.4% for ResNet50), the approach involved transmitting screenshots to a central server for processing and training, thereby compromising user privacy.

Imler et al. explored the use of screen recording combined with image recognition to detect user behavior patterns. However, this approach faced issues related to:

- Storage and processing of large volumes of data,
- Invasiveness of constant screen capture,
- User resistance due to privacy concerns.

3.3 Privacy-Preserving Learning Methods

To address privacy concerns, researchers have proposed technologies such as:

- **Differential Privacy**: Adds statistical noise to data before processing to preserve privacy but can degrade accuracy.
- **Homomorphic Encryption**: Allows computations on encrypted data, though it introduces significant computational overhead.
- **Secure Multiparty Computation (SMPC)**: Enables joint computation over distributed data without sharing raw values, but has high communication complexity.

4. EXISTING SYSTEM

Existing systems and tools designed for screen activity monitoring are primarily developed for corporate environments and employee surveillance. These systems generally focus on tracking user productivity by logging application usage, capturing screenshots, monitoring URLs, and sometimes even recording keystrokes or screen video. While these tools can provide insights into digital behavior, they are not suitable for educational or privacy-sensitive contexts for several reasons.

4.1 Tools and Technologies

Some of the widely used screen monitoring solutions include:

- **Hubstaff**: Provides time tracking, screenshot capture, activity monitoring, and reporting features.
- **Teramind**: Offers user behavior analytics, screen recording, and policy enforcement.
- **DeskTime**: Monitors computer usage time, application categories (productive vs. unproductive), and generates productivity reports.
- **ActivTrak**: Tracks usage patterns, provides productivity scores, and detects anomalies.

4.2 Limitations of Existing Systems

While these systems are robust in terms of features, they exhibit several limitations when evaluated for educational environments:

1. **Privacy Violation**:
   These tools often collect raw data such as screenshots, screen recordings, and browsing history, which may contain private or sensitive information (e.g., emails, passwords, personal chats). This practice violates privacy norms and can lead to data breaches.
2. **Centralized Data Storage**:
   Most monitoring solutions rely on uploading data to centralized servers or cloud systems. This architecture creates a single point of failure and increases the risk of unauthorized data access.
3. **Lack of Content Understanding**:
   These systems cannot differentiate whether a user is watching educational or entertainment content on platforms like YouTube. As a result, they cannot accurately categorize user intent.

4. **Resource Consumption**:
   Continuous screen recording or high-frequency screenshot capture consumes significant system resources and storage space, making the system inefficient, especially on lower-end devices.

5. **Lack of Feedback Mechanisms**:
   Users are typically unaware of their own activity distribution. These systems are designed for supervisors or managers rather than for self-regulation and awareness, which is crucial in education.

4.3 Incompatibility with E-Learning Needs

The above-mentioned drawbacks highlight a critical mismatch between existing systems and the needs of e-learning environments. In educational settings:

- Students must be encouraged to develop self-discipline rather than being heavily monitored.
- Tools must preserve privacy, especially when used on personal devices.
- The system must distinguish productive and unproductive screen use intelligently, even within the same platform.

5. PROPOSED SYSTEM

To address the limitations of existing monitoring tools, this project proposes a novel system for on-screen activity tracking and classification that is both intelligent and privacy-preserving. The proposed system leverages **deep learning for classification** and **federated learning for decentralized model training**, eliminating the need to transmit sensitive user data to a central server.

5.1 System Overview

The system performs the following key functions:

- **Periodic Screenshot Capture**: Captures a screenshot of the user's screen once every minute during usage.
- **Local Activity Classification**: Each screenshot is processed using a lightweight deep learning model (e.g., MobileNetV2) to determine whether the activity falls under:
  - Programming
  - YouTube (Educational)
  - YouTube (Entertainment)
- **Activity Logging**: Each classified activity is logged locally along with a timestamp to maintain a record of user behavior throughout the day.
- **Interactive Dashboard**: The system includes a visually interactive web-based dashboard (using HTML, CSS, JS, and Chart.js) to display the breakdown of time spent in each activity category. Users can observe their screen usage patterns and assess productivity levels.
- **Federated Learning Integration**: Instead of training the model on centralized data, each device trains on its own local data and sends only the **model weights** (not raw images) to a central server. The server aggregates updates to form an improved global model, ensuring data privacy.

5.2 Key Features

- **Privacy Preservation**:
  - No screenshots or raw data are transmitted.
  - Training and inference are performed entirely on the user's device.
- **Content-Aware Classification**:

  - Screenshots are analyzed using image-based classification to distinguish between productive and non-productive activities, even on overlapping platforms like YouTube.

- **Real-Time Feedback**:
  - A local dashboard gives users live insights into their activity patterns.
- **Lightweight and Efficient**:
  - The system uses a lightweight CNN architecture (MobileNetV2) for classification.
  - Screenshot capturing and inference are optimized to reduce system resource usage.

5.4 Advantages over Existing Solutions

- Ensures **complete user privacy** by eliminating data transmission.
- Offers **real-time classification** of screen content.
- Allows users to **self-monitor** productivity rather than being externally surveilled.
- Scalable and adaptable to **personal or institutional** deployments.

6. OBJECTIVES

The primary objective of this project is to design and implement a **privacy-preserving on-screen activity tracking and classification system** tailored for e-learning environments. The system should intelligently monitor and classify screen content into predefined categories, enable self-assessment through an interactive dashboard, and maintain complete data privacy using federated learning techniques.

6.1 Main Objectives

1. **To capture on-screen activity at regular intervals**
   Automatically capture screenshots every minute during system operation without user intervention.
2. **To classify screen activity using deep learning**
   Apply a trained image classification model to determine whether the captured screen content is educational or entertainment-based.
3. **To maintain user privacy by using federated learning**
   Train machine learning models locally on user devices and share only model parameters with a central server, not the data itself.
4. **To provide users with an interactive dashboard**
   Build a real-time interface to display categorized activity summaries, allowing users to self-monitor and regulate their screen usage.
5. **To store activity logs securely on the user's device**
   Keep timestamped activity records locally to allow retrospective analysis without exposing any sensitive data externally.
6. **To support model improvement through decentralized training**
   Aggregate model updates from multiple devices to improve classification accuracy across diverse users without collecting raw user data.

6.2 Additional Goals

- To create a lightweight and efficient system that works seamlessly on personal devices with limited computational resources.
- To design a modular system architecture that can be extended to support additional activity categories or use cases in the future.
- To ensure ethical, non-invasive tracking methods that respect user consent and transparency.

7. SYSTEM REQUIREMENTS

The development and execution of the proposed privacy-preserving screen activity tracking and classification system require specific hardware and software components. These requirements are designed to ensure that the system runs efficiently while remaining lightweight enough for deployment on everyday personal computing devices, such as student laptops.

7.1 Hardware Requirements

Component	Minimum Requirement
Processor	Intel Core i5 / AMD Ryzen 5
RAM	4 GB (8 GB recommended)
Storage	2 GB available disk space
Display	Standard display (HD or higher)
GPU	Not required (GPU optional for training acceleration)
Input Devices	Keyboard, Mouse

7.2 Software Requirements

Component	Specification
Operating System	Windows 10/11, Ubuntu 20.04+, or macOS
Programming Language	Python 3.8 or above
Libraries/Frameworks	Flask, TensorFlow, Keras, NumPy, Pillow, OpenCV, scikit-learn, Chart.js
Web Technologies	HTML5, CSS3, JavaScript (for frontend)
Browser	Chrome, Firefox, or any modern browser
IDE	VS Code, PyCharm, Jupyter Notebook (optional)
Package Manager	pip (Python package manager)

7.3 Python Dependencies (from requirements.txt)

- Flask
- tensorflow
- opencv-python
- numpy
- matplotlib
- Pillow
- scikit-learn
- gunicorn (for production deployment)
- chart.js (included via CDN for frontend visualization)

8. METHODOLOGY

The methodology for this project follows a structured pipeline that integrates **data preparation**, **deep learning model training**, **federated learning**, and a complete **web-based system implementation** (frontend and backend). Each phase of development has been meticulously designed to ensure accurate classification, privacy-preserving learning, and a user-friendly interface for interaction and monitoring.

1. Dataset Preparation

The foundation of the classification model is built upon a carefully curated dataset. The dataset preparation process involved the following key steps:

a. Data Collection:

- Screenshots were collected and categorized into three distinct classes:
    - **Programming**: Images related to code editors, IDEs, development environments.
    - **YouTube_Education**: Screenshots from educational YouTube videos (tutorials, courses, academic content).
    - **YouTube_Entertainment**: Screenshots of entertainment videos from YouTube (music videos, vlogs, comedy, etc.).

b. Data Augmentation:

To improve the generalization of the model and prevent overfitting, data augmentation techniques were applied. This process synthetically expands the dataset by generating altered versions of the original images. Techniques used include:

- **Rotation**: Randomly rotating images to simulate various angles of screen capture.
- **Flipping**: Horizontal and vertical flips to introduce diversity.
- **Zooming**: Zoom-in/zoom-out to simulate different screenshot scales.

This augmentation helped in creating a robust model that can accurately classify a variety of screenshots from slightly different contexts and devices.

2. Model Training

To achieve accurate classification while keeping the computational footprint low, a transfer learning approach was employed using the **MobileNetV2** architecture.

a. Transfer Learning with MobileNetV2:

- **MobileNetV2**, a lightweight convolutional neural network designed for mobile and edge devices, was selected due to its efficiency and accuracy.
- The model was initialized with **pre-trained weights from ImageNet**, allowing it to inherit a strong feature extraction capability.

b. Fine-tuning the Model:

- The top (fully connected) layers were replaced with custom layers suited for the three output classes.
- The base layers were initially frozen to retain generic features from ImageNet.
- The final training phase involved **fine-tuning selected deeper layers** with the collected screenshot dataset to adapt to the domain-specific features.

3. Federated Learning Approach

To enhance privacy and scalability, **federated learning (FL)** was integrated into the training pipeline.

a. Client-Server Architecture:

- FL allows training of models across multiple decentralized clients (e.g., user devices) without sharing raw data.
- Each client trains the model locally on its own dataset and sends **updated weights** (not data) to the central server.

b. Model Aggregation:

- The server collects the weight updates and performs **model aggregation** (e.g., Federated Averaging).
- This aggregated model is then redistributed to all clients for the next round of training.

c. Advantages:

- **Data privacy**: Screenshots remain on the client device.
- **Bandwidth efficiency**: Only model weights are transmitted.
- **Scalability**: New clients can join the training process without retraining from scratch.

4. Backend Development (Flask)

The backend of the system was implemented using the **Flask** web framework. It handles routing, model inference, data logging, and dashboard support.

a. API Routes and Functionalities:

- /classify: Accepts image uploads and returns classification results in real-time.
- /dashboard: Provides data to the frontend for visualization and analysis.

b. Model Integration:

- The trained model is loaded and used within the classification route.
- TensorFlow/Keras handles inference, ensuring fast response times.

5. Frontend Development

The frontend was developed using **HTML, CSS, and JavaScript**, with an emphasis on clean design and interactive visualizations.

a. Real-time Screenshot Classification Interface:

- Users can upload a screenshot or drag-and-drop into the interface.
- The interface sends the image to the Flask backend and receives the predicted class in real-time.
- The result is displayed immediately along with the confidence score.

b. Dashboard & Visualization:

- **Chart.js** is used to visualize user activity data.
- Graph include:
    - Bar charts comparing usage on different days.

6. Real-Time Tracking and Monitoring

An additional feature of the system is **real-time tracking** of the user's activity based on screenshot classifications.

- Screenshots captured periodically (or on specific triggers) are analyzed in real time.
- The system continuously updates usage statistics on the dashboard.

- This allows users (or educators/parents in monitored setups) to monitor content engagement.

9. IMPLEMENTATION

The implementation of the system consists of multiple integrated components, ranging from automated screenshot capturing to real-time classification, backend services, and a user-friendly dashboard interface. Each module is designed to operate seamlessly with the others, providing a complete pipeline from data capture to visualization. The following subsections describe each component in detail:

1. Screenshot Capturing

To monitor user activity and collect visual input for classification, a screenshot capturing mechanism was developed using Python.

a. Tools Used:

- **Python Imaging Library (PIL)**: Used for image manipulation and screen capture.
- **pyautogui / Pillow**: Utilized to capture the full screen of the user's device.

b. Frequency:

- The system is configured to **capture a screenshot every 60 seconds**.
- Screenshots are stored temporarily and then passed to the classifier module for real-time analysis.

c. Automation:

- The screenshot capturing script runs in the background as a scheduled task or a persistent Python process.
- Screenshots are timestamped and automatically managed to avoid memory overload.

2. Screenshot Classifier

The core of the system is a **TensorFlow-based Convolutional Neural Network (CNN)** designed to classify the content of screenshots into one of three predefined categories:

- **Programming**
- **YouTube_Education**
- **YouTube_Entertainment**

a. Model Structure:

- Based on **MobileNetV2**, with pre-trained weights from ImageNet.
- Final layers were customized for the three target classes.
- Dropout and dense layers were added to improve classification robustness.

b. Inference Pipeline:

- Captured screenshots are preprocessed (resized, normalized) before being fed to the model.
- The model returns:
    - Predicted label (class)
    - Confidence score for each category

c. Real-time Execution:

- The classifier processes screenshots as soon as they are captured, ensuring minimal delay between capture and result generation.

3. Flask Server (Backend)

A lightweight and efficient **Flask server** serves as the backend of the application. It handles interactions between the classifier, frontend dashboard, and the screenshot capture module.

a. Key Functional Routes:

- /classify: Accepts POST requests containing image data, performs classification, and returns the result in **JSON format**.
- /get_dashboard_data: Supplies the frontend with aggregated data for visualizations.

b. Technologies:

- **Flask + Gunicorn** (optional) for scalable deployment.
- **JSON-based communication** between frontend and backend.

c. Security Considerations:

- Input validation and file type checks implemented.
- CORS enabled to allow cross-origin requests from frontend.

5. Federated Learning Setup

To ensure privacy and scalability, the system simulates a **federated learning environment** using the **Flower (FL)** framework.

a. Simulation Architecture:

- Multiple **client folders** are created to simulate different devices or users.
- Each client:
  - Has its own local dataset (subsets of screenshots).
  - Trains a local model independently.
  - Sends updated model weights to the central server.

b. Federated Learning Cycle:

1. **Server initializes** global model.
2. **Clients receive** the model and train locally.
3. **Clients return** updated weights to the server.
4. **Server aggregates** the weights (e.g., using Federated Averaging).
5. **Updated global model** is sent back to clients.

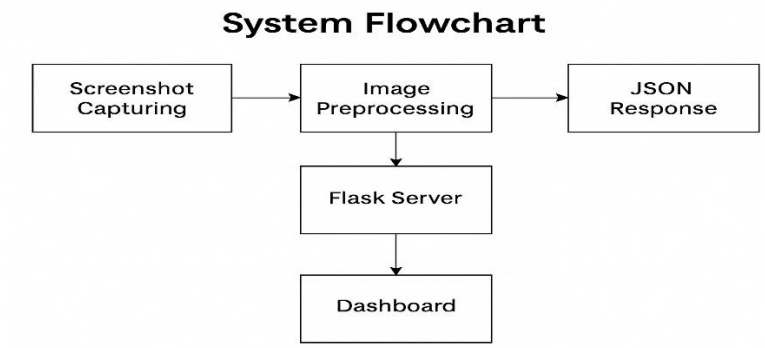
c. Framework:

- **Flower (flwr)**: A flexible framework for building and simulating federated learning systems.
- Easily integrates with TensorFlow and PyTorch.

d. Benefits:

- **Privacy**: Clients never share raw data.
- **Personalization**: Models can be tailored to individual usage patterns.

- **Scalability**: Easily extendable to real-world deployment across multiple edge devices.



10. RESULTS

The system was rigorously evaluated for performance, usability, and privacy-preserving effectiveness. The following results demonstrate the success of the implemented methodology across all components — model accuracy, local classification, user feedback, and dashboard utility.

1. Classification Accuracy

The screenshot classification model achieved **approximately 96% accuracy** on the test dataset, which consisted of unseen samples from all three categories:

- **Programming**
- **YouTube_Education**
- **YouTube_Entertainment**

Key Highlights:

- The model consistently distinguished between educational and entertainment content on YouTube, even with similar visual elements like video players and thumbnails.
- Programming interfaces (e.g., code editors, IDEs) were accurately identified due to the model's fine-tuning on domain-specific visual patterns.
- Augmentation techniques and transfer learning with **MobileNetV2** played a significant role in achieving high performance despite limited data.

2. On-Device Privacy and Federated Learning

One of the standout achievements of this project is its commitment to **data privacy**:

- **All screenshots were processed locally** on the user's device.
- No raw images were transmitted to external servers, thereby protecting sensitive on-screen content.
- Model updates were shared via **federated learning (simulated using Flower)**, ensuring privacy while improving global model performance collaboratively.

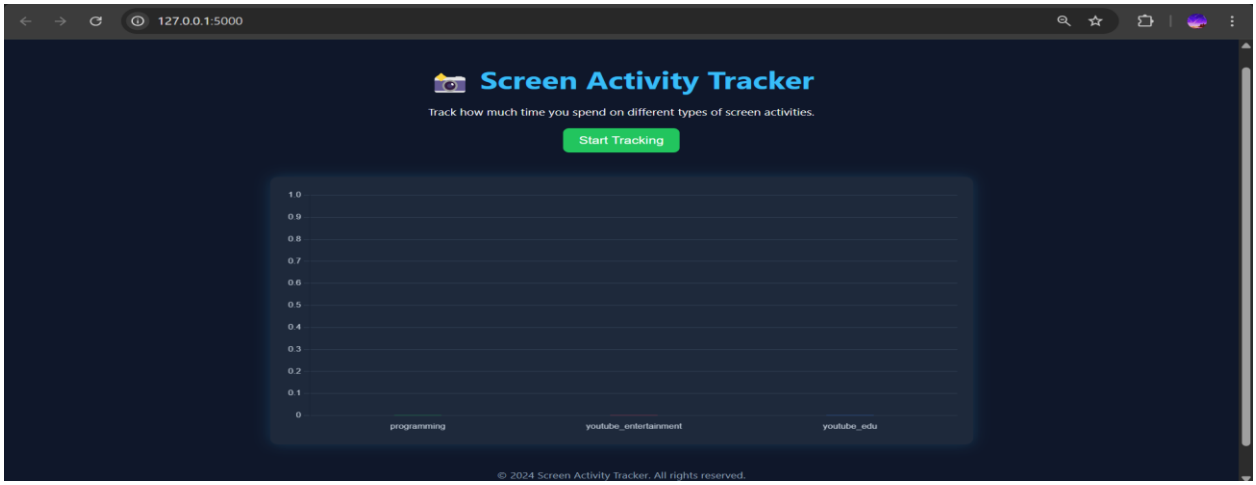
3. Real-Time Logging and Seamless Classification

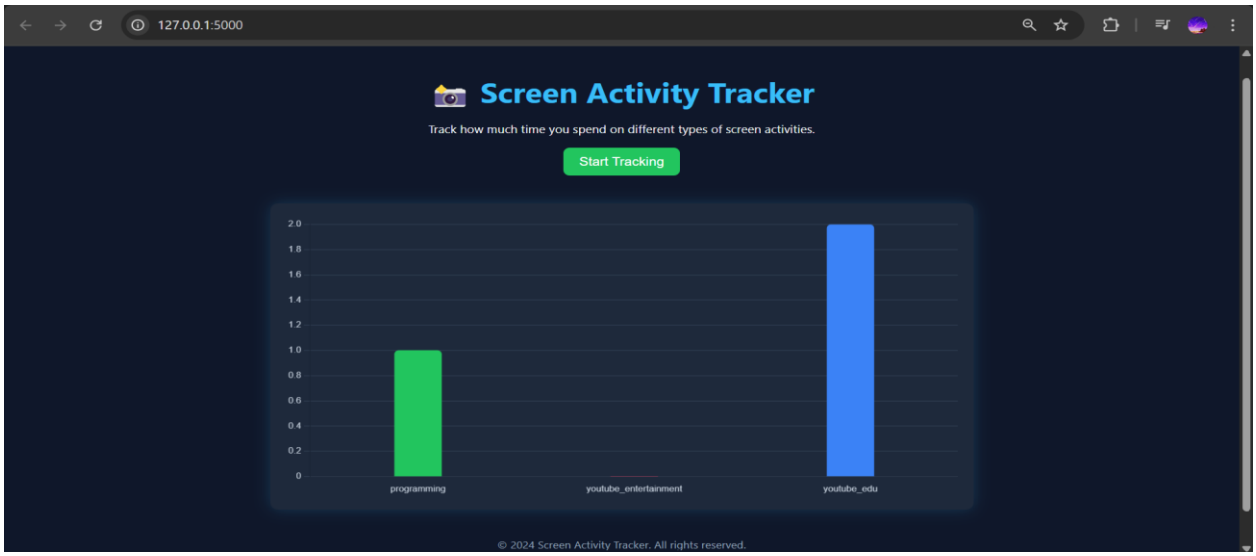
The system successfully:

- Captures screenshots every minute.
- Classifies them in real time using the trained CNN model.

- Logs results (category + timestamp) locally via the Flask backend.

Snapshots





11. CONCLUSION

This system offers a reliable, intelligent, and privacy-preserving solution for monitoring screen activity and assessing e-learning engagement. By capturing screenshots at regular intervals and classifying them into categories such as Programming, YouTube_Education, and YouTube_Entertainment using a lightweight CNN based on MobileNetV2, the system achieves high accuracy (~96%) while running efficiently on local devices. A key strength lies in its commitment to user privacy—no raw data leaves the device, and model training is enhanced using federated learning with the Flower framework, ensuring that user information remains confidential. The real-time, interactive dashboard built with HTML, CSS, JavaScript, and Chart.js allows users to visualize their screen time distribution clearly, enabling them to identify distractions and self-regulate usage. With no reliance on third-party tools, minimal system requirements, and full local deployment, this solution is practical for students, professionals, and educators seeking insights into digital habits without compromising personal data. Ultimately, this project demonstrates how responsible AI and federated learning can be effectively combined to deliver meaningful, privacy-first user experiences in modern screen monitoring applications.

12. FUTURE SCOPE

- **Emotion and Engagement Detection**:
  Integrate facial expression or emotion recognition using webcam input to assess user attentiveness during e-learning sessions.
- **Expanded Classification Categories**:
  Add more activity types such as social media, online meetings, document editing, or gaming for a broader understanding of screen usage.
- **Advanced Time-Based Analytics**:
  Introduce features like hourly trends, daily summaries, and heatmaps to help users visualize productivity patterns over time.
- **Real-World Federated Learning Deployment**:
  Move beyond simulated clients to implement federated learning across actual distributed devices in schools, offices, or homes.
- **Mobile Device Compatibility**:
  Develop a version compatible with smartphones and tablets to monitor screen time across multiple platforms.

REFERENCES

1. **Howard, A. G., et al.** (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. arXiv preprint arXiv:1704.04861.https://arxiv.org/abs/1704.04861
2. **Flower Framework Documentation** – A Friendly Federated Learning Framework. https://flower.dev
3. **TensorFlow** – An end-to-end open-source platform for machine learning. https://www.tensorflow.org
4. **Flask** – Python micro web framework documentation. https://flask.palletsprojects.com
5. **Chart.js** – Simple yet flexible JavaScript charting library. https://www.chartjs.org
6. **Pillow (PIL Fork)** – Python Imaging Library. https://python-pillow.org
7. **Kairouz, P., et al.** (2021). *Advances and Open Problems in Federated Learning*. Foundations and Trends® in Machine Learning, 14(1–2), 1–210. https://arxiv.org/abs/1912.04977
8. **HTML, CSS, and JavaScript Documentation** – MDN Web Docs. https://developer.mozilla.org

9. Dr. Shameem Akther, Saniya Iram Khan, "Federated Learning Acomprehensive study on decentralized Machine Learning", International journal of creative research thoughts,Journal of Scientific Research and Technology, JSRT, Vol. 3 Issue-3, 26/03/2025 Page no. 12-20, ISSN 2583-86660. linkhttps://jsrtjournal.com/index.php/JSRT/article/view/183/222

10. Ms. Sadiya Ansari, Shameem Akther "Dynamic Access Security Protocol with Adaptive Cryptography for Secure Cloud Data Sharing" YMER, Scopus indexed, ISSN : 0044-0477, VOLUME 24 : ISSUE 03 (Mar) – 2025 page no 266-277, March 2025.

11. Mrs. Sana Samreen, Dr.ShameemAkther, " Grid Based Routing for Energy Efficient Data Transmission in IoT – Based WSN", YMER,Vol 24, issue 02, Feb 2025, ISSN-0044-0477 open access, peer reviewed, Scopus active 2025, care UGC Group-II Journal.

12. Mrs. SanaSamreenDr.ShameemAkther " A Survey and Findings to Improve Energy and Reliability for Provisioning Internet of Things Applications using Wireless Sensors Network" YMER, **H-index=5**, Vol 24, issue 02, Feb 2025, ISSN-0044-0477.

13. MsSania Fatima Dr. Shameem Akther "Melanoma detection using Egret search golden optimization – deep convolution neural network model", Elsevier Bio medical signal processing and control, accepted on 6/07/2024

14. Ms. Sadiya Ansari, Dr. Shameem Akther " A Comprehensive survey of data classification techniques and emerging encryption technologies ", YMER, Vol 23, issue 06, June 2024, ISSN-0044-0477 open access, peer reviewed, Scopus active 2024, care UGC Group-II Journal,

15. Dr. Shameem Akther, Syed Fasiuddin, Md. Himayatulla, SujanaAnujum, Shabnam Shadani, "Employing packet analysis for DDos attack detection and prevention" MuktShabd Journal Vol XIII, Issue IV, April 2024, Pg no 1897-1900, ISSn No :2347-3150 Impact factor 4.6

16. Summaiya Yasmeen, Dr.ShameemAkther, " YOLOV5, SignSense: Empoweting Deaf and Mute Communnication through Guesture Recognition", Journal of Scientific Research and Technology, Vol 1, Issue 6, 2023, DOI 10.5281/ zenodo.8332158, date:10/09/2023, ISSN:2583-8660, peer reviewed multidisciplinary journal & recognized by government of India under UDYAM-KR-15-0018523, MSME.

17. SaniaFatima, dr. Shameem Akther, "Skin Cancer using Image Processing" MuktShabd Journal Vol XI, Issue X, October 2022, Pg no 658 – 663, ISSN No : 2347 – 3150 Impact factor 4.6,

18. Ameena Firdous Nikhat, Dr. Shameem Akther, "Effective Heart Rate Estimation from PPG Signal By Fuzzy Wavelet Approach", Webology Volume18, No. 6, Page no. 1800-1815 year 2021 Issn : 1735-188X year 2021

19. Dr. Sameena Banu, Shaista Shireen, "Safe Guarding the Financial Transactions with the Aid of Blockchain Technology", International Journal of Innovative Research in Technology,[IJIRT], November 2021, ISSN: 2349-6002, Vol-8, Issue-6, pp: 318-322, https://www.academia.edu/61766687/.

20. Dr. Sameena Banu, Bibi Hajra Umme E Hani, "Mobile Finger Print Verification and Automatic Log in Platform Using Blockchain", International Journal of Research in Applied Science and Engineering Technology [IJRASET], November 2022, ISSN: 2321-9653, Vol-10, Issue-11, pp: 737-741, http://doi.org/10.22214/ijraset. 2022.47256.

21. Dr. Sameena Banu, Syeda Ummayhani, "Text Summarization and Translation Across Multiple Lanuguages", Journal of Scientific Research and Technology [JSRT], September 2023, ISSN: 2583-8660, Vol1, Issue-6, pp: 242-247,
DOI: 10.5281/zenodo.8370917.

22. Mrs. Farha Naaz, Dr. Sameena Banu, "Input/Output Optimization Schedular for Cloud based Map Reduce Framework", Indonesian Journal of Electrical Engineering and Computer Science, Scopus Indexed (Q3), ISSN: 2502-4752, Vol-35 No. 3, September 2024, pp: 1765-1772, DOI: 10.11591/ijeecs.v35.i3.

23. Mrs. Farha Naaz, Dr. Sameena Banu, "A Multi-core Makespan Model for Parallel Scientific Workflow Executionin Cloud Computational Framework", IAES International Journal of Artificial Intelligence, Scopus Indexed (Q2), ISSN: 2252-8938, Vol-13, No. 4, December 2024, pp: 3849-3857, DOI: 10.11591/ijai.v13.i4.

24. Mrs. Farha Naaz, Dr. Sameena Banu, " A Comprehensive Survey of MapReduce and its Application: Advantages, Challenges, and Research Opportunities", YMER , Scopus Indexed, ISSN: 0044-0477, Vol-23, Issue 11, November 2024, pp: 621-631.
25. Mrs. Nikhat Fatima, Dr. Sameena Banu, "Analysis of Visual Media Alteration to Prevent Frauds using deep Learning Methods", International Journal of Engineering and Science Invention(IJESI), p-ISSN: 2319-6726, Vol-11 Issue-10, Oct' 2022, Impact factor: 5.96, UGC Approval Serial Number: 2573 & UGC Journal Number: 43302.
DOI:10.35629/6734-11106977, pp: 69-77.
26. Mrs. Nikhat Fatima, Dr. Sameena Banu, "The Emerging of Deepfake Media and All-Inclusive Study of technological advances and Media Integrity Threat Challenge", IOSR Journal of Computer Engineering (IOSR-JCE), p-ISSN: 2278-8727, Vol-25, Issue-4, August-2023, pp: 60-68, DOI: 10.9790/0661-2504026068.
27. Mrs. Nikhat Fatima, Dr. Sameena Banu, "Analysis on Splicing and Deepfakes in Viual Forgery and Image Fraud", International Journal of Advance Research and Innovative Ideas in Education, (IJARIIE), UGC CARE, ISSN(O): 2395-4396, Vol-9, Issue-6, 2023, pp: 2712-2019.
28. Mrs. Nikhat Fatima, Dr. Sameena Banu, "Attention-Guided Discrepancy Analysis for Robust Deepfake Detection", YMER, Scopus, ISSN: 0044-0477, Vol-24, Issue-01 January-2025. Pp: 1018-1026.
29. Mrs. Rati D Joshi, Dr. Sameena Banu, " Bio-Inspired Wire Sensor Network: A protocol for an Enhanced Hybrid Energy Optimization Routing", Indonesian Journal of Electrical Engineering and Computer Science, Scopus Indexed, ISSN: 2502-4752, Vol-35 No. 3, September 2024, pp: 1808-1816, DOI: 10.11591/ijeecs.v35.i3.
30. Mrs. Rati D Joshi, Dr. Sameena Banu, "Swarm Intelligent Based Energy Optimization Protocol for Hybrid Routing in Wireless Sensor Networks", International Journal of Engineering, Technology and Applied Research and Systems, Scopus Indexed, Vol-15, Issue-3, pp:23177-23182, June 2025, DOI: 10.48084/etasr.10550 .