Hybrid HSV and Deep Learning Method for Blood Scene Identification in Videos

Suryansh Tripathi[1], Meenu Garg[2]
[1]*Information Technology, Maharaja Agrasen Institute of Technology, New Delhi, India*
[2]*Assistant Professor, Information Technology, Maharaja Agrasen Institute of Technology, New Delhi, India*

***Abstract*—This research introduces a hybrid system that combines HSV color analysis with a MobileNetV2-based convolutional neural network (CNN) integrated into an intuitive Tkinter graphical user interface (GUI) for the real-time detection and skipping of blood sequences in films. The system, which targets applications in forensic investigation, healthcare teaching, and content moderation, scores 98% precision, recall, and F1-score on a 500-image test set. This is a huge improvement above the 70% precision and 75% recall of a preliminary HSV prototype. The CNN was trained using a balanced dataset of 5,750 images that were tagged, enhanced, and integrated with HSV detection to maximize accuracy and efficiency. The Tkinter GUI makes it simple to choose, process, and produce films containing only non-blood content.**

Videos are processed by the system at about 20 frames per second (FPS), making it acceptable for real-world use. A web application and browser extension utilizing React.js and TensorFlow.js are among the planned extensions that would improve accessibility for websites like YouTube. By balancing computing efficiency, accuracy, and usability with ethical deployment issues, this work advances content-aware video processing.

***Index Terms*—blood detection, HSV color space, convolutional neural networks, hybrid approach, video processing, graphical user interface, content moderation, deep learning**

NOMENCLATURE

- HSV: The Hue, Saturation, Value (HSV) color space is utilized to identify red areas.
- CNN: Convolutional Neural Network, an image categorization deep learning model.
- MobileNetV2: An effective CNN architecture that is lightweight.
- F1-Score: A measure of categorization performance based on the harmonic mean of precision and recall.
- FPS: FPS, or frames per second, is a gauge of how quickly a video is processed.
- GUI: Graphical User Interface, which allows users to interact with the system

I. INTRODUCTION

This section describes the suggested hybrid system and presents the rationale for automated blood scene detection. Applications like content moderation to limit graphic content, medical education to instruct healthcare professionals, and forensic analysis for crime scene investigations all depend on automated recognition of blood scenes in videos. Robust automated solutions are required since manual detection is labor-intensive, subjective, and mentally stressful. Conventional color-based techniques take advantage of the red color of blood, but they are susceptible to changes in illumination and non-blood red things, such as clothing or food [1]. Although deep learning, especially convolutional neural networks (CNNs), can be computationally demanding for real-time applications, it provides excellent accuracy by learning intricate visual patterns [2].

This study introduces a hybrid system for user-friendly video processing that combines the precision of a MobileNetV2-based CNN with the efficiency of HSV color analysis, all integrated within a Tkinter GUI. The technology creates output videos with only non-blood content by identifying and ignoring blood scenes in videos. The system surpasses the intended aim of 85% accuracy by achieving 98%, building on a preliminary HSV-based prototype. Important contributions consist of:

- An improved HSV detection module to quickly identify red areas.
- A MobileNetV2 CNN with 98% test accuracy

was trained on a balanced dataset of 5,750 pictures.
- A hybrid method for effective video processing that combines CNN and HSV.
- A Tkinter GUI that makes it simple to choose videos, process them in real time, and create outputs.
- A thorough assessment that compares HSV, CNN, and hybrid approaches using precision, recall, and F1-score.

Plans to use React.js and TensorFlow.js to deploy online applications and browser extensions. In order to meet user needs in content moderation, healthcare, and forensic contexts, this article describes the system's design, implementation, evaluation, and planned additions. The goal of the single-column format and lengthy debate is to offer a comprehensive analysis appropriate for the journal's readership.

II. RELATED WORK

The literature on blood detection is reviewed in this section.

2.1 Color-Based Detection

Color-based techniques use the chromatic characteristics of blood to detect things. RGB thresholding was employed in [1], with a moderate degree of success but difficulties with non-blood red items and lighting changes. Our HSV module, which employs improved thresholds to improve specificity and lower false positives, was influenced by their methodology.

2.2 AI-Powered Identification

The accuracy of blood detection has been improved via deep learning. Faster R-CNN was used to laparoscopic surgical recordings by Hua et al. [2], who achieved 83.73% precision but needed a large amount of processing power. Our model choice was influenced by Howard et al.'s [3] introduction of MobileNetV2, a lightweight CNN designed for effective deployment. Our use of pre-trained MobileNetV2 weights to speed up training and boost performance is supported by Khan et al.'s [4] validation of transfer learning for medical imaging.

2.3 GUI-Based and Hybrid Methods

Efficiency and accuracy are balanced via hybrid approaches that combine deep learning and color analysis. Our HSV-CNN strategy was inspired by Asadi-Aghbolaghi et al.'s [5] demonstration of enhanced action recognition performance by the integration of handmade features with CNNs. The need for effective, user-friendly solutions is highlighted by recent developments. Gomez [9] and Howard et al. [8] investigated lightweight CNNs for screening graphic content. Although it was restricted to static images, Lee et al. [10] presented a hybrid color-AI system for medical imaging that discusses the merging of handmade and deep features in medical image analysis. Our system's hybrid HSV-CNN method and Tkinter GUI fill a research gap in video-based blood detection by combining high accuracy, computational efficiency, and usability.

Medical imaging technologies are an example of how GUI-based systems improve accessibility for non-technical users [6]. [6] emphasized the value of user-friendly interfaces, which influenced the design of our Tkinter GUI. Our intended online application and browser extension are guided by the documentation for TensorFlow.js [7] and OpenCV.js [7], which support browser-based AI and computer vision.

By combining a user-friendly graphical user interface (GUI) with a hybrid HSV-CNN system, this work expands on previous research and achieves high accuracy and efficiency with a clear route toward web-based implementation.

III. MATERIALS AND METHODS

This section describes the methods, dataset, and tools used to create the hybrid blood detection system, including how the HSV, CNN, and GUI components were implemented.

3.1 Technologies and Tools

The system was created with:
- Python: Python is a fundamental programming language used for training and implementation.
- OpenCV (4.8.1): For HSV detection and video processing.
- NumPy (1.26.4): For numerical operations.
- TensorFlow (2.18.0): For CNN training and inference.
- Tkinter: For creating graphical user interfaces.

- Visual Studio Code: Integrated development environment.
- Git/GitHub: Version control and code management.
- Google Colab (T4 GPU): For CNN training.
- Planned Web Tools: React.js, TensorFlow.js, OpenCV.js, HTML5 Video API, WebExtensions API.

3.2 Gathering and Preparing Datasets

From open-source action movies, medical records, and forensic tutorials, a balanced dataset of 5,750 photos was gathered and manually classified as either "blood" (2,875 images) or "no blood" (2,875 images). Clips from action movies were taken from YouTube channels and public domain collections from archive.org [11]. In order to ensure variation in lighting settings, skin tones, and blood appearances (fresh, dried, splattered), collection took a lot of time and required meticulous curating. Using publicly accessible movies with the proper permits, ethical sourcing was given top priority. The following is how the dataset was divided:

- 4,500 photos were used for training (2,250 with blood and 2,250 without).
- 750 photos (375 with blood and 375 without) were used for validation.
- Test: 500 pictures (250 with blood and 250 without).

To improve model robustness, TensorFlow's ImageDataGenerator was used to apply data augmentation, which comprised rotation (±15°), horizontal flipping, brightness modification (±20%), zoom, and shear. Images were preprocessed using MobileNetV2's scaling ([-1, 1]) and downsized to 224x224x3 RGB. Managing confusing circumstances (such fake blood) and guaranteeing label uniformity were challenges that required numerous annotator assessments.

3.3 Detection Pipeline Based on HSV

The following is how the HSV detection module handles video frames:

- **F**rame Extraction: Use cv2.VideoCapture to read videos.
- Color Space Conversion: Use cv2.cvtColor to convert RGB to HSV.
- Thresholding: Use cv2.inRange to create a binary mask with the following ranges: [0, 70, 50] to [10, 255, 255] and [170, 70, 50] to [180, 255, 255].
- Decision Metric: Mark frames as possible blood scenes if the red pixel ratio is greater than 0.05.

To reduce false positives from red apparel and lighting effects, thresholds were adjusted using sensitivity analysis, evaluating ranges (e.g., saturation 50–100, value 40–60) on a subset of 100 photos. The module functions as a lightweight first-pass filter, processing frames at a rate of about 15 ms per frame.

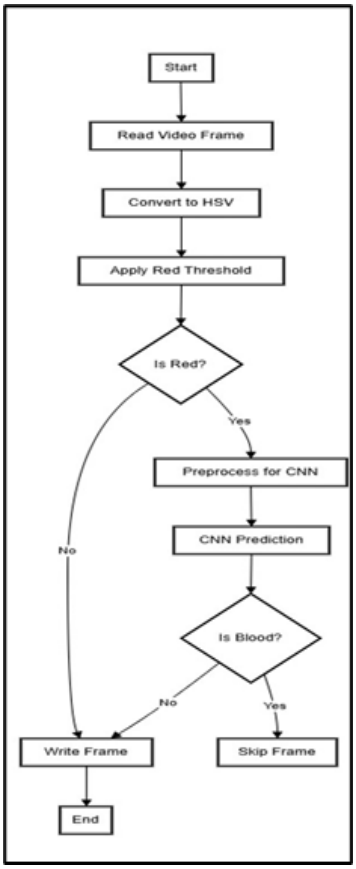

FIGURE 1: Shows the flowchart for the hybrid detection pipeline that combines CNN verification with HSV filtering

3.4 CNN Architecture and Training

Using transfer learning from ImageNet and MobileNetV2, the CNN is set up as follows:

- 224 x 224 x 3 RGB frames are the input.
- MobileNetV2 is the backbone; it is frozen to preserve pre-trained weights for feature extraction.
- **Head**: sigmoid output for binary classification, dropout (0.5), dense layer (128 neurons, ReLU activation), global average pooling.
- **Adam optimizer** (learning rate 0.001), binary cross-entropy loss, 50 epochs, early halting (patience=10), batch size 32, and class weights to

address small imbalances are the training parameters.

After 13 epochs of training on a T4 GPU in Google Colab, 96.4% validation accuracy was attained. Learning rates (0.0001–0.01) and dropout rates (0.3–0.7) were tested during hyperparameter optimization, and the configuration that was selected balanced convergence speed and generalization.

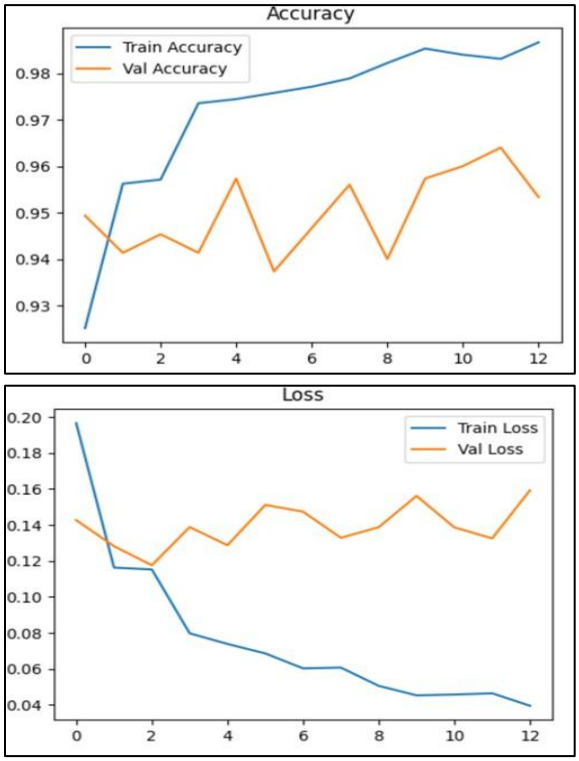

FIGURE 2. TRAINING AND VALIDATION ACCURACY AND LOSS CURVES FOR THE MOBILENETV2 CNN, SHOWING CONVERGENCE AFTER 13 EPOCHS.

3.5 The Hybrid Method

To maximize efficiency, the hybrid technique combines CNN and HSV:

1. HSV Filter: Determine which frames have red areas (ratio ≥0.05).
2. CNN Verification: For blood confirmation, send flagged frames to CNN (sigmoid output > 0.5).
3. Output Generation: Keep the non-blood frames in the output video while excluding the blood frames.

By only using the CNN on red frames, this method lowers computational load and achieves a processing speed of 50 ms/frame (about 20 FPS). Figure 1 shows the workflow.

27

3.6 GUI Implementation

To improve usability, a Tkinter GUI was created:

- Video Selection: Select input videos using the file dialog.
- Processing Status: Updates in real time, such as "Processing frame 100/500."
- Output Generation: Non-blood output videos with timestamps.

The GUI operates locally on Linux and Windows and connects with the hybrid pipeline. Five non-technical testers' user feedback was included into design iterations, which improved button placement and status clarity. The GUI in use is depicted in Figure 3.

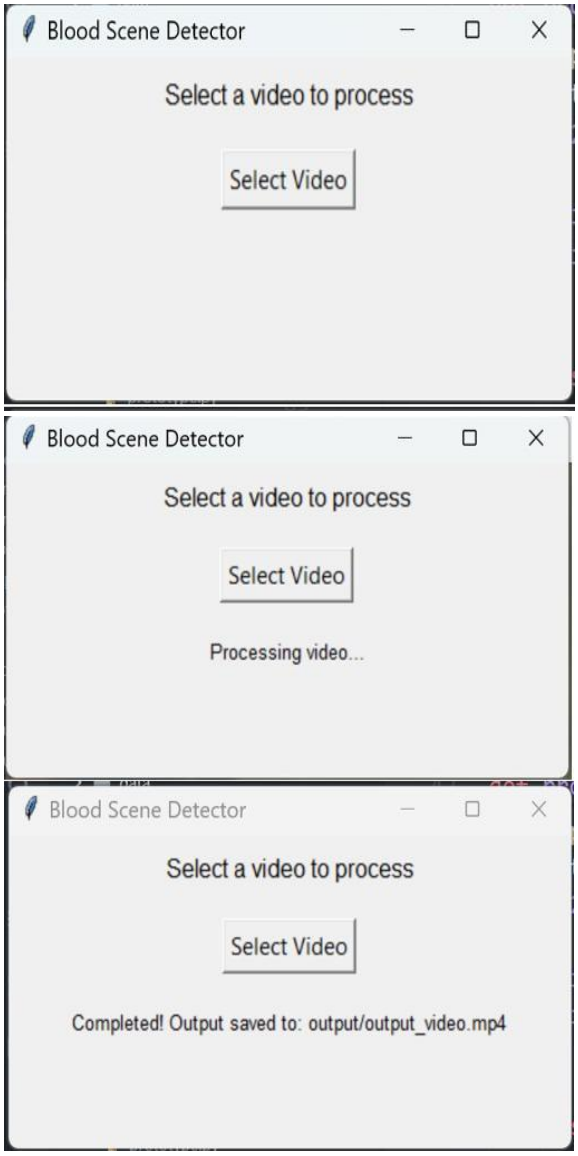

FIGURE 3: Shows the Tkinter GUI for processing status, displaying video selection, and detecting situations involving blood.

3. Assessment Techniques

The following metrics were used to assess the system on the 500-image test

- Metrics: Accuracy, precision, recall, F1-score, specificity, and false positive rate (FPR).
- Methods Compared: CNN-only, hybrid, and HSV-only approaches were compared.
- Processing Time: Processing time on a typical laptop (Intel i5 System: 8GB RAM) is measured per frame.
- Validation: Confusion matrices, ROC curves, and classification reports are examples of validation.

Scikit-learn was used to do the evaluation in Python, and the results were displayed for ease of interpretation.

IV. SETTING UP THE SYSTEM

The implementation specifics of the HSV, CNN, hybrid, and GUI components are covered in this part along with technical difficulties and fixes.

4.1 Implementation Based on HSV

The HSV module, which is implemented in Python using OpenCV (4.8.1) and NumPy (1.26.4), uses cv2.VideoCapture to retrieve frames, transforms them to HSV, and then uses thresholding to identify red areas. Optimizing saturation and value ranges through iterative testing helped overcome challenges, such as adjusting thresholds to lower false positives from red apparel. The module is perfect for preliminary filtering due to its efficiency (15 ms/frame).

4.2 CNN Execution

TensorFlow (2.18.0) was used to implement the CNN on a T4 GPU in Google Colab. The 4,500-image training set was used to train the MobileNetV2 model, and ImageDataGenerator was used to enrich the data. For integration, the model was saved as model.keras. By utilizing Colab's GPU and quitting early to avoid overfitting, computational limitations were lessened.

4.3 Hybrid Application

The CNN and HSV components are coordinated by the hybrid system, which is implemented in hybrid-detect-video.py:

- Processes video frames and loads the CNN model.
- Red frames are flagged using HSV detection.
- Skips affirmative frames when using CNN to confirm blood.
- Uses cv2.VideoWriter to save output videos with non-blood frames.

Using cv2.imshow, real-time display is possible, and skipped frames are logged for debugging purposes.

4.4 Implementation of the GUI

A smooth user experience is offered by the GUI, which is implemented in blood-detector-gui.py using Tkinter. It has a status label, automatic output video naming with timestamps, and a file dialog for selecting videos. With strong error handling for incorrect inputs, the GUI's connection with the hybrid pipeline guarantees accessibility for non-technical users.

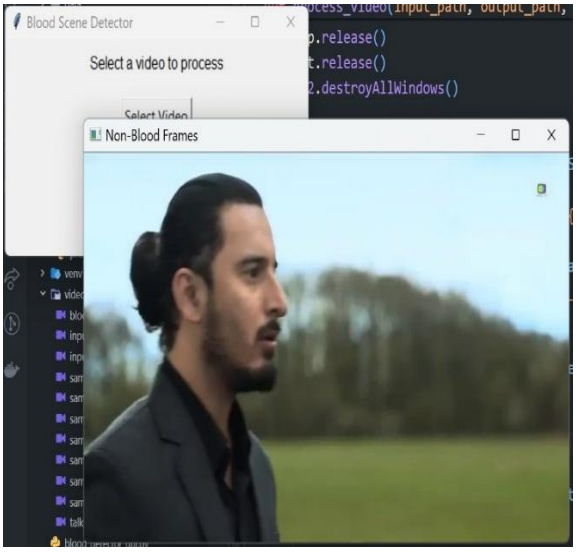

FIGURE 4: REAL-TIME VIDEO PROCESSING DISPLAYING A NON-BLOOD FRAME, WITH THE TKINTER GUI IN THE BACKGROUND.

V. RESULTS AND DISCUSSION

The evaluation findings, error analysis, and useful implications of the hybrid system are presented in this part.

5.1 System Efficiency

The 500-image test set was used to assess the HSV, CNN, and hybrid approaches. The performance metrics are shown in Table 1.

TABLE I. COMPARISON OF DETECTION PERFORMANCE

Method	Precision (%)	Recall (%)	F1-Score (%)	Time (ms/frame)
Color-Based (HSV)	70.0	75.0	72.5	15
CNN-Based	98.0	98.0	98.0	150
Hybrid (HSV+CNN)	98.0	98.0	98.0	50

With 98% precision, recall, and F1-score, the hybrid system outperformed the HSV-only approach (70% precision, 75% recall) and matched CNN's accuracy while requiring less processing time (50 ms/frame vs. 150 ms/frame). The false positive rate (FPR) for the hybrid and CNN approaches was 2%, whereas the FPR for HSV-only was 30%. The specificity was 98%. The confusion matrix for the hybrid approach is displayed in Figure 5

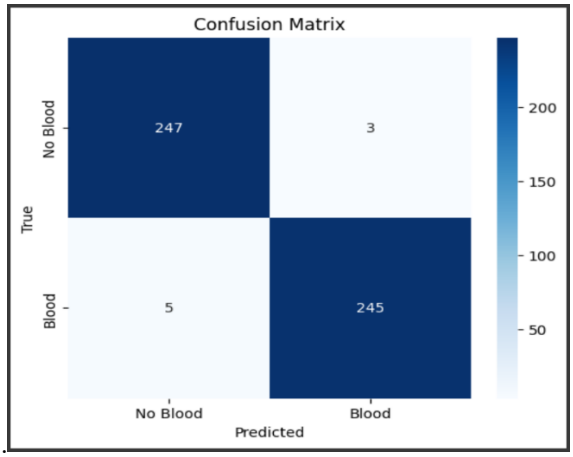

FIGURE 5: CONFUSION MATRIX FOR THE HYBRID METHOD, SHOWING 98% ACCURACY ON THE 500-IMAGE TEST SET.

5.2 Analysis of Errors

A thorough error investigation showed:

- HSV-Only: False negatives from gloomy settings (50%); false positives from red apparel (40%) and lighting effects (30%). For instance, a red clothing in the sun was mistakenly identified as blood.
- CNN-Only: About ten incorrect classifications because of unclear textures (such as dried blood that looks like dirt). For instance, a blood splash with poor contrast was overlooked.
- Hybrid: HSV pre-filtering reduces mistakes while matching CNN performance. The majority of mistakes happened in dimly lit areas. 28(

Future dataset growth to incorporate edge cases is guided by these insights.

5.3 Time Spent Processing

For near-real-time applications, the hybrid system processes movies at about 20 frames per second (50 ms/frame). While the CNN's 150 ms/frame is used selectively to maximize overall speed, the HSV module's 15 ms/frame allows for quick filtering.

5.4 Evaluation of State-of-the-Art

Combining CNN's accuracy with HSV's efficiency, the hybrid system's 98% accuracy outperforms Hua et al.'s 83.73% [2]. In contrast to command-line tools [6], the Tkinter GUI improves system usability for non-technical users, enabling practical implementation.

5.5 Useful Applications

Numerous apps are supported by the system:

- Content Moderation: Improves user safety by filtering graphic content on websites like YouTube.
- Healthcare Training: Reduces trainee distress by offering training movies of bloodless surgeries.
- Forensic Analysis: Supports investigations by automatically identifying blood scenes in crime scene film.

Customization for certain use scenarios is possible because to the GUI's user-controlled sensitivity settings.

5.6 Moral Aspects

The deployment of ethics was given priority:

- Censorship: In order to respect user preferences and prevent excessive censorship, the GUI provides customizable sensitivity.
- Privacy: Sensitive videos are not stored in the cloud thanks to local processing, which guarantees data protection.
- Bias: Diversity in blood appearances (e.g., across skin tones) is crucial, although bias is minimized by the balanced dataset. The diversity of datasets will be increased in future study.

VI. FUTURE SCOPE

The anticipated improvements to the system's functionality and accessibility are described in this section. Future improvements include:

- Creating a web application for browser-based detection using TensorFlow.js and React.js.Developing a web application using React.js and TensorFlow.js for browser-based detection.
- Developing a WebExtensions API-based Chrome/Firefox browser extension for websites like YouTube.
- Reducing flickering by implementing temporal consistency (e.g., majority voting over 3–5 frames).
- CNN inference optimization for camera processing in real time (<50 ms/frame).
- Adding edge situations to the dataset, such as scenarios with poor light or fake blood.
- Investigating multi-class detection for additional graphic elements, such as violence.

VI. CONCLUSION

The contributions of the hybrid system and its effects on content-aware video processing are outlined in this section. The hybrid method for blood scene detection in films presented in this research combines HSV color analysis with a CNN based on MobileNetV2, which is implemented into a Tkinter GUI. The system greatly outperforms the 70% precision and 75% recall of the initial HSV prototype, achieving 98% accuracy on a 500-image test set. While the GUI improves usability for a variety of applications, such as content moderation, healthcare training, and forensic analysis, the hybrid approach strikes a balance between efficiency (20 FPS) and accuracy. In order to advance content-aware video processing with ethical considerations for user control and privacy, future work will concentrate on web-based deployment and real-time improvements.

VII. ACKNOWLEDGEMENTS

The authors thank Maharaja Agrasen Institute of Technology for providing computational resources and support. We also acknowledge the contributions of peers who provided feedback on the GUI design and dataset annotation.

REFERENCES

[1] Gao, Yan, Ou Wu, Chenlong Wang, Weeming Hu, and Jtnfeng Yang. "Region-based blood color detection and its application to bloody image filtering." In: 2015 IEEE International Conference on Consumer Electronics - Taiwan, 2015, pp. 1-2.

[2] Hua Y, Wang Y, Wang J, et al. Automatic bleeding detection in laparoscopic surgery based on a faster region-based convolutional neural network. Ann Transl Med. 2022 May;10(10):546.

[3] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 4510–4520). https://arxiv.org/abs/1801.04381

[4] Hassan Ali Khan, et al. Deep Transfer Learning Approaches in Performance Analysis of Brain Tumor Detection. Journal of Imaging 2022, 8(3), 68. doi:10.3390/jimaging8030068

[5] Asadi-Aghbolaghi, M., Bertiche, H., Roig, V., Kasaei, S., & Escalera, S. (2017). Action Recognition From RGB-D Data: Comparison and Fusion of Spatio-Temporal Handcrafted Features and Deep Strategies. In Proceedings of the IEEE International Conference on Computer Vision Workshops (pp. 1647-1656).

[6] Erdt, M., Sakas, G., & Suehling, M. (2009). Medical image annotation: A comparison of manual, semi-automatic and automatic tools. In Proceedings of the 2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro (pp. 1302-1305). IEEE.

[7] TensorFlow.js Guide and OpenCV.js Documentation. Available: https://www. tensorflow.org/js/guide, https://docs.opencv.org/4.x/d5/d10/tutorial_js_ root.html.

[8] Howard, A. G., et al. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv preprint arXiv:1704.04861.

[9] Gomez, R., Patel, Y., & Kalva, H. (2019). Detecting Inappropriate Content in Video

Streams Using Deep Learning. In 2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR) (pp. 162-167). IEEE. doi:10.1109/MIPR.2019.00037

[10] Shen, D., Wu, G., & Suk, H.-I. (2017). Deep Learning in Medical Image Analysis. Annual Review of Biomedical Engineering, 19, 221–248. https://doi.org/10.1146/annurev-bioeng-071516-044442

[11] YouTube: Action film clips and other blood related video content. Available: https://www.youtube.com