**Detection of Violent Scenes in Cartoon Videos**

### ****Project Title:** Detection of Violent Scenes in Cartoon Videos**

### ****Submitted By****

* **Name**: Muhammad Bilal Sajid
* **Roll No.**: SU92-BSAIM-F23-018
* **Section**: BSAI 3A

### ****Submitted To****

* **Instructor Name**: Sir Rasikh Ali

### ****Abstract****

This project implements a deep learning-based system to detect violent scenes in videos. The model processes video frames to classify them as either violent or non-violent and applies a blurring effect to violent scenes. The system ensures safe video content by marking and logging violent sections with timestamps. The project leverages Convolutional Neural Networks (CNNs) for classification, supported by custom Python scripts for preprocessing, training, and video processing.

### ****Introduction****

The increasing consumption of digital video content necessitates tools to identify and manage violent content. This project addresses the challenge by providing an automated solution to detect and handle violent scenes in videos. It employs deep learning techniques to classify frames and processes them for further action, such as blurring violent scenes.

Key objectives:

1. Detect violent content in video frames with high accuracy.
2. Blur violent frames to ensure content suitability.
3. Log timestamps and confidence levels for violent scenes.

### ****Methodology****

### ****Dataset****

The dataset used in this project was downloaded from [Kaggle](https://www.kaggle.com/datasets/mohamedmustafa/real-life-violence-situations-dataset). It contains labeled video samples divided into two categories:

1. **Violent**
2. **Non-Violent**

Frames were extracted from the video samples for training and validation purposes.

#### ****1. Dataset Preparation****

* **Frame Extraction**:  
  Video frames are extracted from labeled video datasets (violent and non-violent) using OpenCV.
* **Labeling**:  
  Frames are labeled based on their parent video category (violent = 1, non-violent = 0).

#### ****2. Model Architecture****

The system uses a Convolutional Neural Network (CNN) with the following key components:

* **Convolutional Layers**: Extract features from input frames.
* **MaxPooling Layers**: Downsample feature maps to reduce computation.
* **Dropout Layers**: Prevent overfitting during training.
* **Dense Layers**: Perform classification.

#### ****3. Training the Model****

* **Data Augmentation**: Increases data diversity by applying transformations like rotation, zoom, and horizontal flips.
* **Model Training**: Trained with categorical cross-entropy loss and Adam optimizer over multiple epochs, with validation for performance monitoring.

#### ****4. Evaluation****

* **Accuracy and Classification Report**: Evaluates the model on a validation set.
* **Confusion Matrix**: Analyzes classification errors.

#### ****5. Video Processing****

* **Sliding Window Technique**: Analyzes consecutive frames to identify violent content.
* **Blurring**: Applies a Gaussian blur effect to violent frames.
* **Logging**: Records frame numbers, timestamps, and confidence levels for violent scenes in a CSV file.

### ****Implementation Details****

#### ****Technologies and Tools****

* **Programming Language**: Python
* **Libraries and Frameworks**: TensorFlow/Keras, OpenCV, Numpy, MoviePy
* **Hardware**: Dell Latitude with 16GB RAM, 512GB NVMe SSD
* **Dataset**: Downloaded from [Kaggle](https://www.kaggle.com/datasets/mohamedmustafa/real-life-violence-situations-dataset)

#### ****Scripts Overview****

1. **extract\_frames.py**  
   Extracts frames from videos and saves them in labeled directories.
2. **load\_data.py**  
   Prepares datasets by loading and preprocessing images.
3. **train\_model.py**  
   Trains a CNN model to classify frames as violent or non-violent.
4. **evaluate\_model.py**  
   Evaluates the model using accuracy and classification metrics.
5. **predict.py**  
   Predicts whether a single image is violent or non-violent.
6. **process\_video.py**  
   Processes videos to detect and blur violent scenes, logs timestamps, and outputs a new video.

### ****Results****

**Log Example**

The system generates a CSV file logging details of violent frames. Below is an example log entry:

| **Frame Number** | **Classification** | **Confidence** | **Action Taken** |
| --- | --- | --- | --- |
| 858 | Violent | 98.55% | Blurred |

This entry corresponds to a frame in the input video that was classified as violent with a confidence score of **98%**, and the frame was blurred accordingly.

**Key Results**

* **Input Video**: *VID-20241030-WA0001.mp4*
* **Output Video**: *cartoon\_video\_blurred.mp4*
* **Violent Frames Processed**: All identified violent frames were successfully blurred.

#### ****Sample Outputs****

* Blurred video output: cartoon\_video\_blurred.mp4
* Violent scenes log: violent\_scenes\_log.csv

### ****Challenges and Limitations****

1. **Dataset Size**: Limited availability of violent/non-violent video datasets impacted training diversity.
2. **Real-Time Processing**: Processing large videos requires significant computational resources.
3. **False Positives**: Occasional misclassification of non-violent scenes.

### ****Future Improvements****

1. Incorporate transfer learning with pretrained models for better accuracy.
2. Enhance dataset size and diversity to improve generalizability.
3. Optimize processing speed for real-time applications.
4. Integrate Grad-CAM for model explainability.