Violence Detection in Video

Introduction

The prevalence of violent incidents captured on video has significantly increased with the widespread use of surveillance systems, smartphones, and social media platforms. Detecting violence in videos has become a critical necessity for enhancing public safety, preventing crimes, and enabling rapid response in emergencies. Automated violence detection systems can assist law enforcement agencies, private security firms, and public organizations in monitoring real-time threats, reducing human monitoring fatigue, and enabling proactive interventions.

The use cases for such systems are vast, ranging from surveillance in public spaces like malls, schools, and transport hubs to content moderation on online platforms to prevent the spread of harmful content. Additionally, violence detection systems can be employed in forensic video analysis and assist in maintaining the safety of staff and inmates in correctional facilities.

Traditional methods for violence detection relied heavily on manual monitoring or heuristic-based approaches, which are not only time-intensive but also prone to inconsistencies and inaccuracies. With advancements in computer vision and deep learning, automated systems can now achieve significant accuracy in analyzing video streams. Methods leveraging convolutional neural networks (CNNs), recurrent neural networks (RNNs), and transfer learning have demonstrated the potential to identify violent actions with high precision by analyzing motion patterns, object interactions, and scene contexts.

This project explores the implementation of a deep learning-based approach to detect violence in videos. By combining state-of-the-art models and video preprocessing techniques, the system aims to achieve reliable and efficient violence detection, addressing critical gaps in existing methods while highlighting the potential for real-world applications.

Dataset

The dataset used for this project is taken from Kaggle: "Real Life Violence Situations Dataset". It contains 1000 violent videos and 1000 non-violent videos. The videos are of varying lengths and frame sizes, containing a variety of violent and non-violent situations. The violent situations occur in various settings ranging from public brawls to violent sports, and the non-violent situations occur in normal daily life, sports, dance and music.
Some videos had problems in compression or number of frames which made them unsuitable for use and had to be dropped from the dataset. After which the videos were split into train and test sets in an 80:20 ratio.

Static Model

1. Deep Learning Network Architecture

The deep learning network used in this project for violence detection is based on the **ResNet-50 (Residual Network)** architecture. This network is a widely adopted model for image classification tasks due to its ability to handle deep architectures without the vanishing gradient problem.

ResNet-50 Overview

- **Depth**: 50 layers (consists of convolutional, batch normalization, ReLU activation, and pooling layers).
- **Key Innovation**: The introduction of **Residual Blocks**, which allow the network to learn an identity mapping more effectively, mitigating the degradation problem observed in very deep networks.
- **Architecture Breakdown**:
    - Initial convolution and max pooling layers.
    - 4 stages of residual blocks with increasing depth.
    - Global average pooling layer.
    - Fully connected (FC) layer with softmax activation for final classification.

The ResNet-50 model is pre-trained on the **ImageNet** dataset, allowing for **transfer learning**, which helps in speeding up training and achieving high accuracy even with a smaller dataset.

2. Residual Block and Identity Mapping

The core component of ResNet-50 is the **Residual Block**, which can be expressed mathematically as:

$$y=F(x,\{Wi\})+x$$

- **x**: Input to the block.
- **F(x,{Wi})**: Represents the residual function, which consists of convolutional layers with weights {Wi}\{W_i\}{Wi} and a non-linear activation function (such as ReLU).
- **y**: Output of the block.

The addition of x (identity mapping) ensures that the gradient can flow easily through the network during backpropagation, reducing the vanishing gradient problem.

$$\begin{tikzcd}
\text{Input} \arrow[r] \arrow[rr, bend right, dash] & \text{[Conv} \rightarrow \text{BN} \rightarrow \text{ReLU} \rightarrow \text{Conv} \rightarrow \text{BN]} \arrow[r] & \text{Add} \arrow[r] & \text{ReLU} \arrow[r] & \text{Output}
\end{tikzcd}$$

3. Training Algorithm

The model is trained using the **Cross-Entropy Loss** and the **Adam (Adaptive Moment Estimation)** optimizer. The combination of these techniques ensures efficient training and convergence.

Cross-Entropy Loss

For a binary classification task (violence vs. non-violence), the Cross-Entropy Loss is defined as:

$$L=-(1/N) * \sum_{i=1}^{N} [y_i*\log(\hat{y}_i)+(1-y_i)*\log(1-\hat{y}_i)]$$

N: Number of samples.

y_i: True label (0 for non-violence, 1 for violence).

$\hat{y}_i$: Predicted probability for class 1 (violence).

Adam Optimizer

The **Adam optimizer** is used to update the network weights during training. It combines the benefits of **AdaGrad** and **RMSProp** by adapting the learning rate for each parameter.

4. Training Workflow

1. **Forward Pass**:
   - Input images are passed through the ResNet-50 model to produce class probabilities.
2. **Loss Calculation**:
   - Compute the Cross-Entropy Loss between the predicted and actual labels.
3. **Backward Pass**:
   - Calculate gradients using backpropagation.
4. **Weight Update**:
   - Update the weights using the Adam optimizer.

5. Model Visualization with Grad-CAM Equation

The **Grad-CAM heatmap Lc** for class c is computed as:

$$L^c_{Grad-CAM} = ReLU\left(\underbrace{\sum_k a^c_k A^k}_{\text{Linear combination}}\right)$$

Where:

- **A^k**: Activation map from the target convolutional layer.
- **α^c.k**: Weights computed by performing global average pooling on the gradients of the target class c with respect to the activation map A^k

The **ReLU** function ensures that only positive influences are considered in the heatmap.

This equation helps identify which regions in the input image contribute most to the model's prediction for class c.

Results:

Training Performance:

- The training accuracy improves consistently across epochs, reaching **97.99%** by the third epoch.
- Training loss decreases with each epoch, indicating that the model is learning effectively.

Testing Performance:

- The best test accuracy achieved is **96.00%**, recorded during the first epoch.
- The final test accuracy is **95.14%**, which demonstrates good generalization performance.

Temporal Model

The R3D-18 model is a 3D convolutional neural network designed for video action recognition, extending the 2D ResNet-18 architecture into the temporal domain to effectively capture spatiotemporal features.

1. Deep Learning Network Architecture

- Input Layer: Accepts video clips formatted as tensors with dimensions (B, C, T, H, W), where B is the batch size, C is the number of channels (typically 3 for RGB), T is the number of frames, and H and W are the height and width of each frame.
- Stem: Begins with a 3D convolutional layer using a kernel size of (3, 7, 7), followed by batch normalization and a ReLU activation function. This layer processes the input video clip to extract initial spatiotemporal features.
- Residual Blocks: Comprises four layers of residual blocks, each containing multiple BasicBlocks. These blocks utilize 3D convolutions to capture complex spatiotemporal patterns:
    - Layer 1: 64 filters
    - Layer 2: 128 filters
    - Layer 3: 256 filters
    - Layer 4: 512 filters
- Downsampling: Implemented via convolutions with a stride of 2 at the beginning of each layer (except the first), reducing the spatial and temporal dimensions while increasing the depth of feature maps.
- Pooling and Fully Connected Layer: After the residual blocks, an adaptive average pooling layer reduces the feature map to a fixed size, followed by a fully connected layer that outputs class scores corresponding to the number of action categories.

Key Features:

- 3D Convolutions: Utilizes 3D convolutions throughout the network to simultaneously model spatial and temporal information, enabling effective action recognition in videos.
- Residual Connections: Incorporates skip connections to facilitate gradient flow and mitigate the vanishing gradient problem, allowing for deeper network architectures.
- Pretrained Weights: Pretrained on the Kinetics-400 dataset

Modifications:

- The fully connected classifier head of the original model contains 400 neurons since Kinetics-400 is a 400 class dataset. However, ours is a binary classification problem, hence the final FC layer was updated to contain 1 neuron.

- To directly obtain prediction probabilities from the model, sigmoid activation was added to the output layer.

2. Data Preparation

SInce the model has an input sequence length of 16 frames, the data loader randomly selects a sequence of 16 frames from each video and applies the default transforms provided by PyTorch which are:

1. Resize the frames to 128x171 using bilinear interpolation
2. Central crop with crop size 112x112
3. Rescale the values to be between 0.0 and 1.0
4. Normalize with mean = [0.43216, 0.394666, 0.37645] and standard deviation = [0.22803, 0.22145, 0.216989].

3. Training

The model was trained on 80% of the dataset using the following settings:

- Optimizer: Adam with initial learning of 0.001
- Loss: Binary Cross Entropy loss
- Epochs: 10
- All parameters are frozen except the final fully connected layer

4. Results

The final model achieved an accuracy of 92.77% and an F1 score of 93.18% on the test dataset, demonstrating good generalization and discriminatory strength.

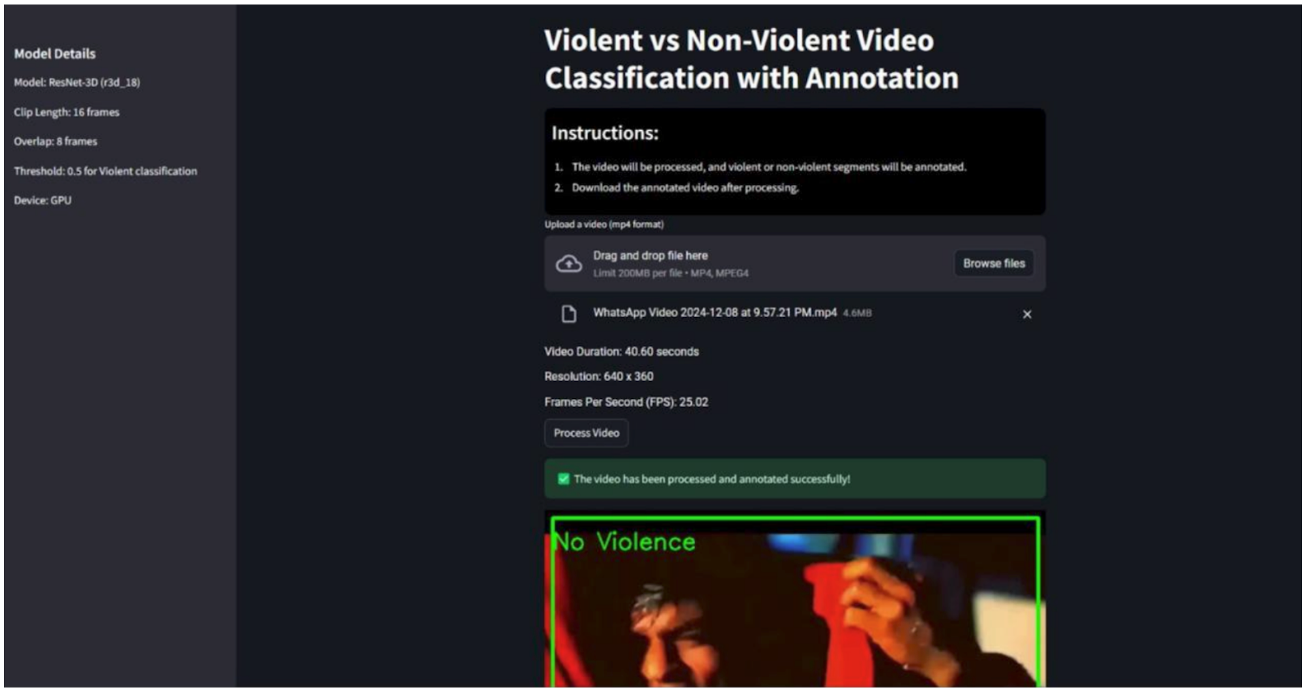
Frontend

The frontend was designed using Streamlit, a Python-based framework for building interactive web applications. The primary focus was on user simplicity and accessibility. Here are the key contributions:

- The interface was developed to provide a user-friendly experience for uploading videos and visualizing results.
- Custom CSS was implemented to enhance the aesthetics, including improved fonts, subtle backgrounds, and responsive elements.
- Clear instructions were added to guide users on how to use the interface effectively.
- The frontend extracts and displays video metadata such as duration, resolution, and FPS (frames per second) to ensure user transparency.

- A loading spinner provides real-time visual feedback during video processing, keeping users informed of the system's status.
- A sidebar is included in the interface to display the technical details of the backend model, such as architecture, clip length, overlap, and device type.
- The interface features a download functionality, allowing users to download the annotated video, adding practicality. This is complemented by clear and actionable error messages to enhance the user experience and assist with troubleshooting efficiently.
- The frontend interacts seamlessly with the backend through well-defined functions, ensuring efficient processing of uploaded videos.

Here’s the output :



Model Details

Model: ResNet-3D (r3d_18)

Clip Length: 16 frames

Overlap: 8 frames

Threshold: 0.5 for Violent classification

Device: GPU

Violent vs Non-Violent Video Classification with Annotation

Instructions:

1. The video will be processed, and violent or non-violent segments will be annotated.
2. Download the annotated video after processing.

Upload a video (mp4 format)

WhatsApp Video 2024-12-08 at 9.57.21 PM.mp4 4.6MB

Video Duration: 40.60 seconds

Resolution: 640 x 360

Frames Per Second (FPS): 25.02

The video has been processed and annotated successfully!

Conclusion

The project effectively implements a deep learning based framework for violence detection in videos, leveraging both static image models and temporal video models. By employing architectures such as ResnET-50 for static analysis and R3D-18 for temporal modeling the project achieves strong performance metrics, validating robustness of chosen methodologies. The incorporation of video annotation offers an intuitive way to visualize and interpret the results.

With further enhancements, this framework could lay the foundation in showcasing potential of deep learning in real world scenarios. It showcases the importance of integrating advanced models with accessible interfaces for efficient and scalable solutions.