Individual Final Report

Name: Raghav Agarwal

GWID: G32909283

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.

Division of Collaborative Work

- @Raghav Agarwal & @Anirudh Rao: Understanding and developing the static model.
- @Dhanush Bhargav & @Guruksha: Developing a model that incorporates delay to evaluate how the sequence of frames over time enhances the model's ability to detect violence in videos.
- @Raghav Agarwal: Analyzing the data by plotting frames with respect to class, creating the training script for the static model, and writing the inference script for the static model.

Individual Contributions by Raghav Agarwal

1. Data Analysis: Analyzed the dataset by visualizing frame distributions with respect to different classes to gain insights into the data.
2. Static Model Training Script: Developed the training script for the static model, ensuring proper implementation and optimization.
3. Static Model Inference Script: Created the inference script for the static model to effectively evaluate its performance on new data.
4. Collaboration on Static Model Development: Worked alongside Anirudh Rao to understand and develop static model architecture.

Description of work in detail

Data Analysis *(plot_V_NV_frames.py)*

Code Description

This script performs data visualization and frame extraction from videos classified into two categories: violence and non-violence. The script follows a structured approach to analyze and visualize the middle frames of randomly selected videos and their frame count distributions.

Key Functionalities

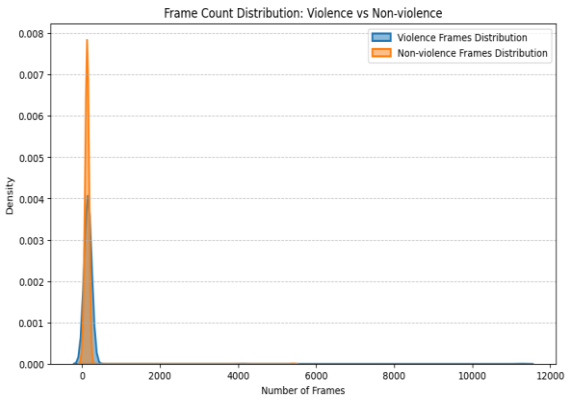
1. Configuration Setup:
    - Reads paths from a config.conf file to access datasets and specify output directories.
    - Ensures the output visualization directory is freshly created for saving results.
2. Random Video Selection:
    - Function: get_random_videos(directory, num_videos=5)
    - Randomly selects 5 videos from each category (violence and non-violence).
3. Frame Count Calculation:

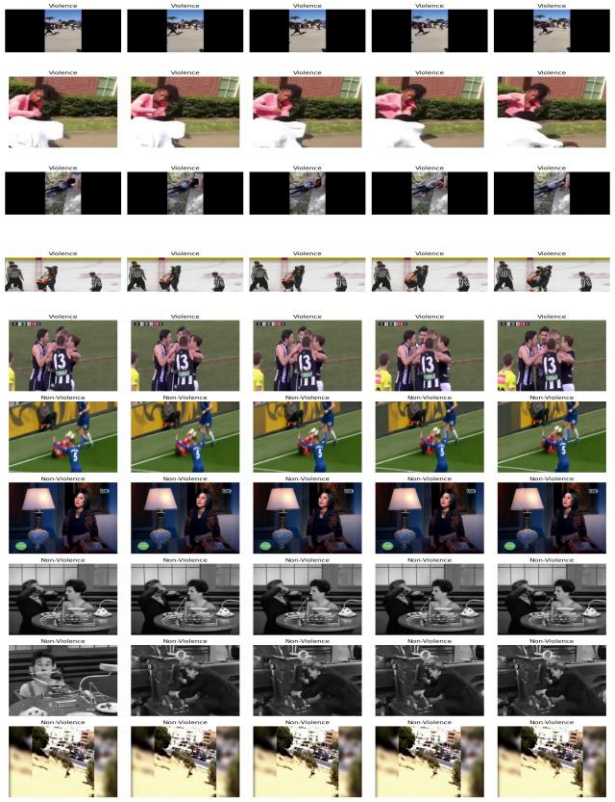
- Function: get_frame_counts(video_path)
- Calculates the total number of frames in each video for both categories.

4. Middle Frame Extraction:
    - Function: extract_middle_frames(video_path, num_frames=10)
    - Extracts 10 middle frames from the selected videos.
5. Frame Count Distribution Visualization:
    - Plots the distribution of frame counts for violence and non-violence videos using seaborn.
    - Saves the plot to the output directory as violence_vs_nonviolence_distribution.png.
6. Frame Visualization:
    - Extracts and plots the middle 5 frames from each selected video.
    - Displays the frames in a 10x5 grid with titles indicating Violence or Non-Violence.
    - Saves the resulting grid plot as frames_grid.png.

Workflow Summary

1. Initialize Configuration:
    - Load dataset paths and set up the output directory.
2. Select Random Videos:
    - Pick 5 random videos each from the violence and non-violence categories.
3. Analyze Frame Counts:
    - Compute and visualize frame count distributions.
4. Extract Middle Frames:
    - Extract middle frames from selected videos.
5. Plot Frames:
    - Display the extracted frames in a grid for visual inspection.





Percentage of code:

Total Lines of code = 165

Taken from internet ~ 30

Added lines of code ~ 135

Modified lines of code ~ 10

Training Static model *(Static-Model-Train.py)*

This script is designed to train, evaluate, and test a violence detection model using a pre-trained ResNet-50 neural network on image frames. The frames are categorized into **violence** and **non-violence** classes, and the goal is to classify images accurately into these two categories.

Key Components and Workflow

1. **Configuration Setup**:
    - Reads paths from config.conf to specify locations of datasets and model output directory.
    - Creates the model output directory if it doesn't exist.
2. **Dataset Definition**:

- **Class**: ViolenceDataset
  - Loads images from the specified directories for violence and non-violence frames.
  - Assigns labels: 1 for violence and 0 for non-violence.
- **Transformations**:
  - Converts images to tensors and normalizes them with mean and standard deviation of 0.5.

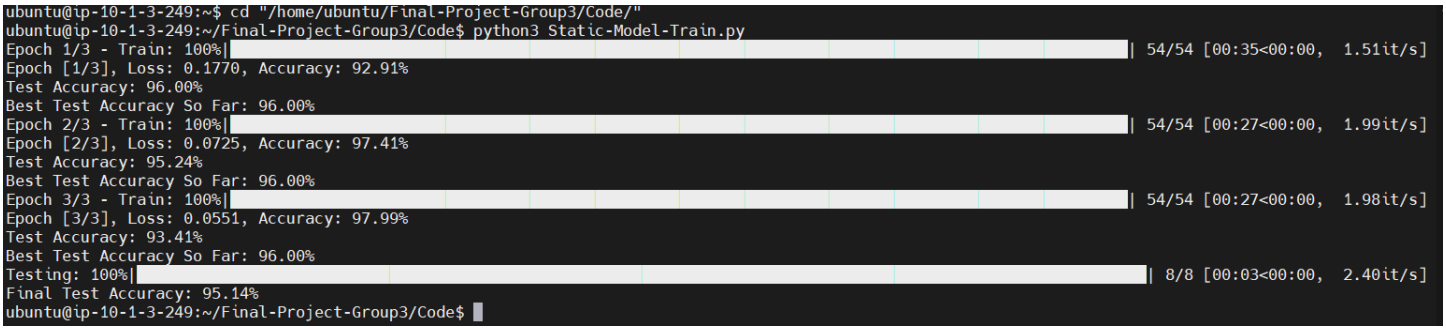
3. **Data Splitting**:
   - Splits the dataset into:
     - **Train Set**: 70%
     - **Test Set**: 20%
     - **Validation Set**: 10%
   - Creates DataLoaders for each subset with a batch size of 128.
4. **Model Architecture**:
   - **Class**: ViolenceClassifier
     - Uses a pre-trained **ResNet-50** model.
     - Replaces the final fully connected layer to output 2 classes (violence and non-violence).
5. **Training the Model**:
   - **Function**: train_model
     - Trains the model using the CrossEntropyLoss and Adam optimizer.
     - Tracks the loss and accuracy during each epoch.
     - Saves the model with the best test accuracy to static_model.pt.
6. **Testing the Model**:
   - **Function**: test_model
     - Loads the best model and evaluates it on the validation set.
     - Outputs the final test accuracy.
7. **Execution Workflow**:
   - **Train the Model**: Trains the model for 50 epochs.
   - **Test the Model**: Loads the saved model and evaluates it on the validation set.

Summary of Functions

- **ViolenceDataset**: Loads image data and assigns labels.
- **train_model**: Trains the ResNet-50 model and saves the best-performing model.
- **test_model**: Evaluates the saved model on the validation set.

Key Points

1. **Pre-trained Model**: Uses ResNet-50 for transfer learning, speeding up convergence and improving performance.
2. **GPU Support**: Moves the model and data to GPU if available.
3. **Performance Metrics**:
    - Accuracy during training and testing.
    - Saves the model with the highest test accuracy.
4. **Progress Bar**: Uses tqdm for tracking training and testing progress.



Percentage of code:

Total Lines of code = 170

Taken from internet ~ 20

Added lines of code ~ 150

Modified lines of code ~ 10

Inference of static model (Inference_static_model.py)

This script applies Grad-CAM (Gradient-weighted Class Activation Mapping) to visualize the regions of video frames that contribute to the model's prediction for violence detection. It processes each

frame of a given video, generates Grad-CAM heatmaps, overlays them on the frames, and creates a GIF from the processed frames.

Workflow Summary

1. Model Definition:
    - Loads a custom ResNet-50 model (ViolenceClassifier) with a modified final layer for binary classification (violence vs. non-violence).
    - The model weights are loaded from the saved checkpoint (static_model.pt).
2. Grad-CAM Initialization:
    - Uses Grad-CAM to identify which parts of an image are most important for the model's prediction.
    - Targets the last convolutional layer (layer4[-1]) of the ResNet-50 model.
3. Preprocessing:
    - Transforms video frames by resizing them to 224x224, converting them to tensors, and normalizing with a mean and standard deviation of 0.5.
4. Video Processing:
    - Reads each frame of the specified video (V_138.mp4) using OpenCV.
    - For each frame:
        - Applies Grad-CAM to generate a heatmap.
        - Overlays the heatmap on the frame.
        - Saves the resulting frame with the Grad-CAM visualization to the output_frames folder.
5. GIF Creation:
    - Collects all the saved frames and creates an animated GIF (gradcam_output.gif), with a frame duration of 100 ms and looping enabled.

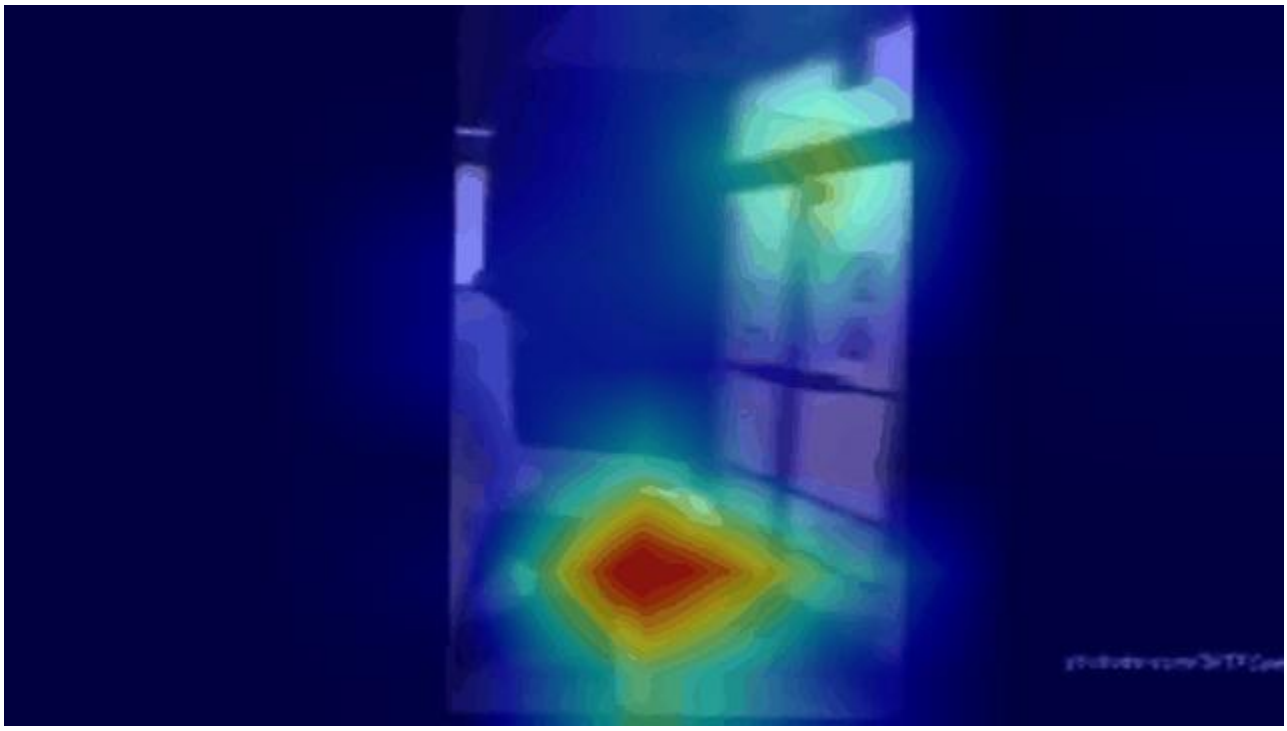
Key Functions and Libraries

- Libraries:
    - OpenCV (cv2): For reading and processing video frames.
    - PyTorch: For model inference and Grad-CAM computation.
    - Grad-CAM: Visualizes model activations to explain predictions.
    - PIL: Creates the final GIF from the processed frames.
- Key Methods:

- GradCAM: Applies Grad-CAM to visualize the model's focus.
- show_cam_on_image: Overlays the Grad-CAM heatmap on the original image.
- cv2.VideoCapture: Reads video frames sequentially.
- Image.save: Saves a sequence of images as a GIF.

Folder and File Outputs

- output_frames: Folder containing processed frames with Grad-CAM overlays.
- gradcam_output.gif: Final animated GIF showing Grad-CAM visualizations for each frame.



Percentage of code:

Total Lines of code = 66

Taken from internet ~ 60

Added lines of code ~ 6

Modified lines of code ~ 0

Summary

This report presents the development and implementation of a deep learning-based violence detection system for video analysis. The project addresses the increasing need for automated violence detection in video streams due to the widespread use of surveillance systems, smartphones, and social media platforms. The primary goal is to enhance public safety, reduce human monitoring fatigue, and enable proactive responses to violent incidents.

The work is divided into three major components:

Data Analysis:

- Visualized frame distributions for violence and non-violence categories.
- Selected random frames and plotted them in a grid to gain insights into the dataset.

Static Model Training:

- Implemented a violence classification model using a pre-trained ResNet-50.
- Trained the model on image frames extracted from video data and optimized it using transfer learning.
- Split the dataset into training, testing, and validation sets to ensure robust performance evaluation.
- Achieved high accuracy by saving the best-performing model based on test results.

Inference and Visualization:

- Applied Grad-CAM to visualize which regions of video frames influenced the model's predictions.
- Generated an animated GIF showing Grad-CAM overlays, providing an interpretable visualization of the model's decision-making process.
- The contributions include the development of scripts for data analysis, model training, and inference, ensuring a comprehensive approach to violence detection.

Conclusion

- The project successfully demonstrates the feasibility of using deep learning for automated violence detection in videos. By leveraging a ResNet-50 model and Grad-CAM for visualization, the system achieves reliable performance and offers interpretability for its predictions. The key outcomes include:
- Efficient Model Training: Achieved high accuracy through transfer learning, optimizing the performance of a static violence detection model.

- Clear Visual Explanations: Grad-CAM overlays provide transparency into the model's decision-making, which is essential for trust and deployment in real-world applications.
- Practical Applications: The system can be deployed in public spaces, surveillance systems, and content moderation platforms to enhance safety and response times.

Percentage of code:

Taken from internet: 110

Modified lines of code: 20

Added lines of code: 291

(110 – 20) / (110 + 291) = 0.22

References:

https://github.com/jacobgil/pytorch-grad-cam

https://pytorch.org/tutorials/beginner/introyt/trainingyt.html