INDIVIDUAL FINAL REPORT

ANIRUDH KUMAR RAO

Introduction

As technology becomes increasingly integrated into daily life, the need for systems that can quickly and accurately identify threats has grown in urgency. Real-time violence detection represents a groundbreaking shift in how we approach safety and security, transforming reactive measures into proactive solutions. From bustling urban centers to online platforms, the ability to monitor and respond to violent activities instantly can redefine the standards of public safety.

The applications of such technology stretch far and wide. In public spaces like stadiums, schools, and train stations, it can provide instant alerts, enabling security personnel to intervene before situations escalate. In the digital realm, it can help curb the spread of violent content across social media, fostering safer online communities. Even in controlled environments like correctional facilities, real-time detection systems can minimize risks to both staff and inmates.

This project explores an innovative approach to real-time violence detection, leveraging advanced deep learning techniques to analyze videos as they are captured. By focusing on efficiency and accuracy, the system aims to not only identify violent incidents but to do so in a manner that is scalable, adaptable, and ready to meet the dynamic challenges of the modern world.

Outline of Shared Work

- Static/Frames Model: Understanding, development and training – Anirudh and Raghav
- Temporal Model: Understanding, development and training – Dhanush and Guruksha
- Inference of Static Model – Raghav
- Inference of Temporal Model – Anirudh and Dhanush
- Streamlit App – Dhanush and Guruksha

My Individual Contribution

1. Data Preprocessing for Static/Frames model
2. Training Pipeline for Static/Frames model – ViolenceClassifier Class (Training done by Raghav)
3. Frame/Clip Annotation for Video Model Inference

Description of Individual Work

1. Data Preprocessing -- Frame_extraction.py

   Overview: This file will access the videos from their respective directories and create two new directories: Violence_frames and NonViolence_frames which is used downstream for training the static model.

   - load_videos_from_directory() function loads the base videos and creates a csv so that we can access the base video path for frame extraction. The paths and their labels are saved as a csv which is accessed in this code for frame extraction

   - extract_frames() function loads the video one by one and captures and saves frames based on the set frame rate. The frame rate is default set to 1 which will give us 1 frame for every second. If more frames are needed we need to change the input frame_rate to 1/frames needed. eg for 4 frames per second, frame_rate = 0.25

   Percentage of Code:

   - Taken from the Internet: 30
   - Modified : 10
   - Added Lines: 66
   - Total Lines: 106

2. Training Pipeline for Static/Frames model – Baseline_CNN.py / Static-Model-Train.py

   Overview: This script is used to train, evaluate and test our violence detection Static model on the Image Frames extracted from the videos using pretrained ResNet50 network. The goal of this script is to understand how well we can accurately classify these images as violent or non violent.

   - Dataset Class: Loads all the Images from the Violence and Non Violence Frame directories and assigns the labels based on which directory the image was retrieved from (1 for violent 0 for non violent).

```
class ViolenceDataset(Dataset):  1 usage  Rag-hav385
    def __init__(self, root_dir, transform=None):  Rag-hav385
        self.root_dir = root_dir
        self.transform = transform

        # Get list of all image paths
        self.violence_images = glob.glob(os.path.join(root_dir, 'Violence', '*.jpg'))
        self.nonviolence_images = glob.glob(os.path.join(root_dir, 'NonViolence', '*.jpg'))
        # Combine lists with labels
        self.image_paths = self.violence_images + self.nonviolence_images
        self.labels = [1] * len(self.violence_images) + [0] * len(self.nonviolence_images)

    def __len__(self):  Rag-hav385
        return len(self.image_paths)

    def __getitem__(self, idx):  Rag-hav385
        image_path = self.image_paths[idx]
        label = self.labels[idx]
        image = Image.open(image_path)

        if self.transform:
            image = self.transform(image)

        return image, label
```

- The loaded images are converted to tensors and transformations(if needed) can be applied. Currently we are just normalizing the tensors to ensure they are bound within [-1,1]

- Classifier: ViolenceClassifier() is mapped to ResNet50 Architecture where we load the pretrained weights and configure the last layer to classify the images according to our needs (num_classes=2)

```
# Load a pre-trained ResNet model
class ViolenceClassifier(nn.Module):  2 usages  Anirudh Rao
    def __init__(self):  Anirudh Rao
        super(ViolenceClassifier, self).__init__()
        # Load a pre-trained ResNet model
        self.model = models.resnet50(pretrained=True)
        # Modify the final layer for binary classification
        num_features = self.model.fc.in_features
        self.model.fc = nn.Linear(num_features, out_features: 2)  # 2 classes: violence and non-violence

    def forward(self, x):  Anirudh Rao
        return self.model(x)
```

- The remaining training was performed by my project member Raghav and the following results were obtained – Test Accuracy 95.14%

```
ubuntu@ip-10-1-3-249:~$ cd "/home/ubuntu/Final-Project-Group3/Code/"
ubuntu@ip-10-1-3-249:~/Final-Project-Group3/Code$ python3 Static-Model-Train.py
Epoch 1/3 - Train: 100%|██████████| 54/54 [00:35<00:00,  1.51it/s]
Epoch [1/3], Loss: 0.1770, Accuracy: 92.91%
Test Accuracy: 96.00%
Best Test Accuracy So Far: 96.00%
Epoch 2/3 - Train: 100%|██████████| 54/54 [00:27<00:00,  1.99it/s]
Epoch [2/3], Loss: 0.0725, Accuracy: 97.41%
Test Accuracy: 95.24%
Best Test Accuracy So Far: 96.00%
Epoch 3/3 - Train: 100%|██████████| 54/54 [00:27<00:00,  1.98it/s]
Epoch [3/3], Loss: 0.0551, Accuracy: 97.99%
Test Accuracy: 93.41%
Best Test Accuracy So Far: 96.00%
Testing: 100%|██████████| 8/8 [00:03<00:00,  2.40it/s]
Final Test Accuracy: 95.14%
ubuntu@ip-10-1-3-249:~/Final-Project-Group3/Code$
```

Percentage of Code:

- Taken from the Internet: 10
- Modified : 0
- Added Lines: 104
- Total Lines: 114

3. Inference Pipeline for ResNet_3D_18 – run_inference.py and inference_and_annotate.py

   Overiew: These scripts are used to generate the video with annotations that follow the temporal sequence of predictions made by the ResNet_3D_18 Model. The final video contains a bounding box and a label that changes colour based on the predictions made for each clip within the video.

   Run_inference is used in the downstream Streamlit to showcase the predictions for any uploaded video.

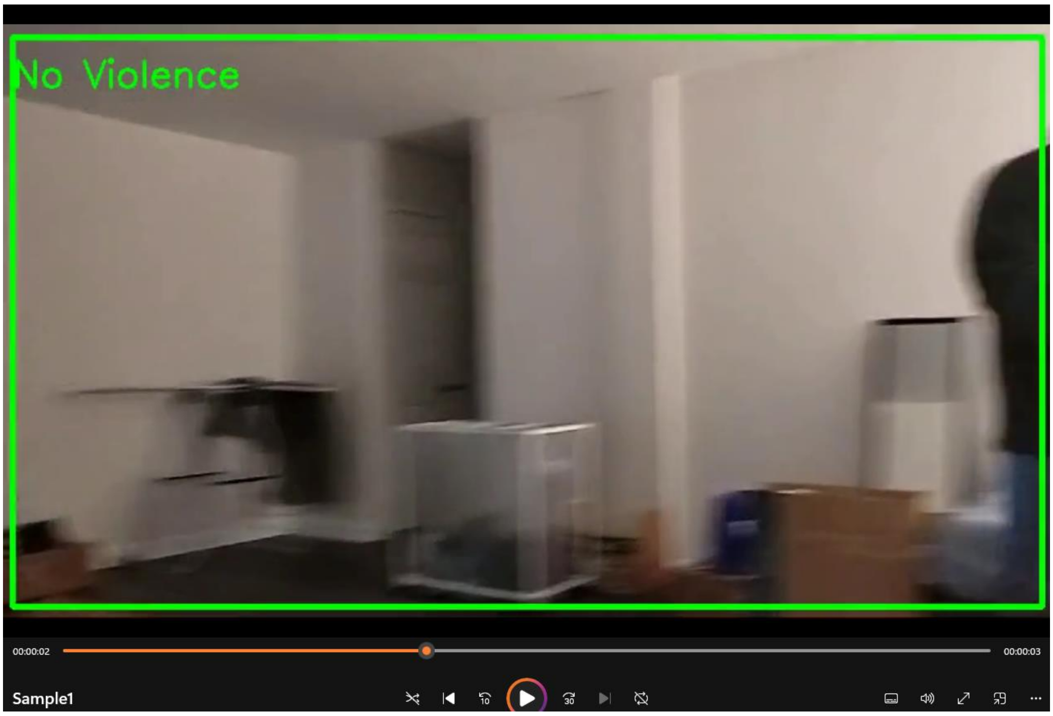
   Inference_and_annotate essentially calls run_inference for each video present in the Test Dataloader to vizualize the model predictions on the test set

   - ViolenceClassifierInference() is our classifier that is loaded with our trained weights. The infer() function within the class returns the predictions made for the supplied clips

```
class ViolenceClassifierInference():  5 usages  dhanush
    def __init__(self, model_path, device):  dhanush
        self.classifier = ViolenceClassifier(num_classes=1)
        self.transforms = R3D_18_Weights.DEFAULT.transforms()
        self.classifier.load_state_dict(torch.load(model_path))
        self.device = torch.device(device)
        self.classifier.to(self.device)
        self.classifier.eval()

    def infer(self, video_frames):  3 usages (1 dynamic)  dhanush
        input = self.transforms(video_frames)
        input = input.to(self.device)
        with torch.no_grad():
            output = self.classifier(input)
        output = output.detach().cpu().numpy()
        return output
```

   - Stich_clips_with_annotations() : is called to collect the outputs generated by all the clips and add the label and coloured bounding box to each frame within the clip. It then writes out(stitches) each annotated frame with the help of CV2's VideoWriter. This generates the original video with all its frames along with a dynamic indicator in the video itself highlighting when it is violent or non-violent



Percentage of Code:

- Taken from the Internet: 50
- Modified : 0
- Added Lines: 60
- Total Lines: 110

Summary

The project involves multiple stages, starting with data collection and preprocessing. A curated dataset containing instances of real-life violence and non-violent scenarios was used to train the model.

The system was also integrated with video alert mechanisms to notify relevant personnel when violent incidents were detected through annotation mechanisms. Performance metrics such as accuracy and F1-score were used to evaluate the system's accuracy and reliability.

This project represents a significant step forward in leveraging artificial intelligence to improve public safety. By combining state-of-the-art deep learning techniques the violence detection system offers a scalable and effective solution to the challenges posed by manual video monitoring. Future work could explore expanding the dataset, improving model robustness across different environments, and integrating the system with broader safety and surveillance.

Percentage of All Codes:

- Taken from the Internet: 90
- Modified : 10
- Added Lines: 230
- Total Lines: 330

(110 – 10)/(110+230) = 0.303

Future Scope:

- Additional functionalities to include Real time processing so that it can be deployed to edge devices like CCTV cameras
- The Static model could be used for additional tasks such as blurring violent pictures/videos such as crime scene photos

References

- https://geeksforgeeks.org/saving-a-video-using-opencv/
- https://docs.opencv.org/4.x/dd/d9e/classcv_1_1VideoWriter.html#gsc.tab=0
- https://www.youtube.com/watch?v=AxIc-vGaHQ0
- https://pytorch.org/tutorials/beginner/introyt/trainingyt.html