

A Internship Report On

Real-time Violence Detection in Public Safety Using a ResNet50-Based Video Analytics Framework

Dept. Of CSE,
M S Ramaiah Institute of Technology,
Banglore

Abstract

Human motion analysis has gained significant attention in recent years, emerging as a central focus in video processing, particularly in the domains of motion detection and security surveillance. The impetus behind this research lies in the pursuit of tools that minimize human resources while optimizing specific tasks. Video surveillance has evolved into a cornerstone of modern security systems, offering a mechanism to monitor public spaces and ensure the safety of individuals. In this intricate landscape, the identification of violent and non-violent activities within CCTV footage assumes a pivotal role in advancing public safety measures. This research proposes a sophisticated system designed to harness the potent capabilities of ResNet50 (Residual Neural Network) for the automated detection of both violent and non-violent events in CCTV videos. The overarching goal is to contribute substantially to the development of security solutions that are not only more effective but also highly efficient, facilitating timely responses to potential threats.

TABLE OF CONTENTS

Chapter No.	Title	Page No.
1	INTRODUCTION	1
	1.1 General Introduction	1
	1.2 Problem Statement	3
	1.3 Objectives of the internship	3
	1.4 Internship deliverables	3
2	INTERNSHIP DISCUSSION	4
	2.1 Objectives Achieved	4
	2.2 Scientific and Professional Skills	4
	2.3 Challenges Experienced during the Internship	5
3	METHODOLOGY	6
	3.1 Tools Used	6
	3.2 Literature Survey	7
	3.3 Block Diagram	8
	3.4 Use Case Diagram	10
4	RESULTS	12
	4.1 Software Description	12
	4.2 Results	14
5	CONCLUSION	16
	REFERENCES	17

INTRODUCTION

1. General Introduction

Societies are grappling with various challenges arising from rapid population growth, and one pressing issue is the need to enhance personal security. As a response, the implementation of continuous monitoring in public spaces becomes imperative to bolster security measures and concurrently reduce the societal costs associated with violence. Notably, numerous surveillance cameras surround public places such as shopping malls, airports, and streets, presenting an opportunity for event monitoring. The adoption of real-time violence detection methods within these camera systems can significantly expedite the identification process and promptly alert security authorities, enabling swift intervention.

The complexity of violence detection in research arises from the need to interpret violent actions within specific contexts and situations. Differentiating between actions that should be classified as violence varies based on the context; for instance, kicking in the streets may be deemed violent, while the same action could be considered entirely normal in certain sports like rugby.



Fig 1.1: Violence and Non-violence Detection

Video Analysis with Deep Learning:

Deep Learning-Oriented Video Analysis Deep learning is a type of machine learning that uses neural networks to find patterns in data. Neural networks are composed of layers of interconnected, multi-layered processing nodes. Due to the rapid advancements in deep learning, artificial intelligence has shown to be a highly effective tool for video analysis. Specifically, deep learning algorithms are used to track and detect objects in video as well as to identify certain actions.

Violence and Non-violence Detection in Video Analytics:

Violence and non-violence detection in video analytics stands as a crucial element within contemporary security systems, utilizing advanced computer vision algorithms to discern and track such incidents in monitored areas. Employing sophisticated object detection models, like ResNet50, ensures rapid identification of potential violent or non-violent events while minimizing false positives. Classification algorithms contribute to the precision of this identification process, distinguishing between violent actions and other non-threatening activities for accurate threat assessment.

Integrated tracking mechanisms monitor the movement of individuals engaged in violent or non-violent actions across video frames, providing valuable insights into their behavior and trajectory. Upon detection, alerting mechanisms promptly notify designated personnel, facilitating swift responses to mitigate security risks. Continuous refinement and optimization of the model ensure its adaptability to evolving threats, enhancing effectiveness across various operational environments.

Supplementary sensor technologies further augment surveillance capabilities, fortifying overall threat detection proficiency. In summary, violence and non-violence detection in video analytics play a pivotal role in proactive security measures. This technology empowers organizations to detect and respond to incidents with precision and efficiency, fostering a safer and more secure operational environment.

Beyond detecting violent and non-violent activities, our system integrates advanced video analytics capabilities to analyze and extract meaningful insights from captured video footage. This involves tracking the movement patterns associated with detected violent events, monitoring activity trends over time, and generating actionable intelligence to augment situational awareness and enable more informed decision-making.

1.2 Problem Statement:

Collecting diverse video data from public areas and ensuring compatibility with the ResNet50 architecture pose challenges due to privacy concerns, scalability issues, and environmental variations. Integration of ResNet50 into the video analytics pipeline demands meticulous attention. The model requires fine-tuning with a thoughtfully curated dataset, accurately representing instances of both violent and non-violent behavior. Enhancing violence detection capabilities entails effectively capturing temporal information within video sequences and extracting relevant features through ResNet50 layers. Ensuring reliable inference, minimal latency, and computational efficiency are the primary hurdles in optimizing the ResNet50 model for real-time processing of video streams.

1.3 Objectives of the internship

Hands-on Learning: Internships offer a chance for students or beginners to apply what they've learned in real work situations, bridging the gap between theory and practice.

Skill Building: Interns get to develop a variety of skills, from technical ones related to their field to soft skills like communication and teamwork.

Exploring Career Paths: Internships let individuals try out different careers and industries before committing long-term, helping them figure out what they enjoy and where they want to go in their careers.

Feedback for Growth: Interns receive feedback on their performance, highlighting strengths and areas for improvement. This helps them develop strategies for personal and professional growth.

Project Contributions: Interns often work on specific projects, contributing their skills and ideas to make a positive impact within the organization.

1.4 Internship deliverables

The project aims to create a violence and non-violence detection system for videos using a ResNet50-based deep learning model trained on annotated video datasets. The deliverables include a robust model, utilizing TensorFlow, for accurate classification after extracting features through ResNet50. Data augmentation will be applied for generalization, and OpenCV will be used for video frame extraction. The final outcome will be a trained model capable of effectively distinguishing between violent and non-violent activities in videos.

2. Internship Discussion

2.1 Objectives Achieved

1. Describe the dataset that the ResNet50 model was trained and tested on. Talk about its scope, variety, and applicability to the actual world.
2. Clearly highlight the issue of violence monitoring and the necessity of automated techniques in order to improve public safety and security.
3. Describe the precise types of violence that are the focus of the study in order to establish its scope.

2.2 Skills (scientific and professional):

2.2.1 Scientific skill

1. **Data Analysis:** Proficiency in analyzing diverse datasets, understanding patterns, and extracting meaningful insights to inform the training process. This includes exploratory data analysis (EDA) and statistical assessments to ensure the dataset's quality.
2. **Machine Learning and Deep Learning:** Thorough understanding of deep learning architectures, especially DenseNet, and machine learning principles. To get high-performance results, one must comprehend model architecture, hyperparameter tuning, and optimization strategies.
3. **Computer Vision:** Proficiency in computer vision techniques, enabling efficient extraction of features from video frames. This encompasses a grasp of image processing, object recognition, and segmentation techniques pertinent to video analytics.
4. **Technical Knowledge:** Strong programming abilities, especially with Python, which is a language used for creating and deploying DenseNet models. It is essential to be familiar with pertinent libraries and frameworks like PyTorch or TensorFlow.
5. **Algorithmic Thinking:** Robust algorithmic mindset for crafting efficient algorithms in the identification of violent and non-violent activities, ensuring optimal model performance and scalability.

2.2.2 Professional skill

- 1. Project Management:** Planning, organizing, and executing tasks to meet project milestones and deadlines.
- 2. Team Collaboration:** Working effectively within a team environment to coordinate tasks and share progress.
- 3. Communication:** Clearly communicating project objectives, progress, and results to supervisors and team members.
- 4. Problem-solving:** Identifying and addressing technical challenges encountered during system development.
- 5. Ethical and Legal Considerations:** Understanding the ethical and legal implications of drone detection systems, including privacy concerns and regulatory compliance.

2.3 Challenges Experienced during the Internship

The internship experience in crafting a violence and non-violence detection system using ResNet50 unveiled numerous overarching challenges. A pivotal hurdle involved acquiring and pre-processing a dataset that accurately mirrored real-world scenarios, covering a wide range of situations essential for robust model training. Tackling ethical considerations, especially pertaining to privacy issues associated with video analytics, introduced an additional stratum of intricacy. Despite their demanding nature, these challenges proved instrumental in fostering significant learning experiences and skill enhancement in the realm of computer vision and deep learning within the ResNet50 framework.

3. METHODOLOGY

The proposed system for the internship project on violence and non-violence detection using ResNet50 followed a systematic and phased methodology. To commence, an extensive literature review was conducted to assimilate knowledge on prevailing methodologies and advancements in violence detection, with a specific focus on deep learning architectures like ResNet50. Subsequently, the project's scope and objectives were delineated, specifying the types of violence targeted for detection and addressing ethical considerations associated with system deployment. The dataset, consisting of diverse video clips depicting real-world scenarios, underwent meticulous curation and preprocessing. The implementation of the ResNet50 architecture followed suit. The final methodology reflects a thorough and data-driven approach, ensuring the robust development and evaluation of a violence and non-violence detection system using ResNet50.

3.1 Tools used:

TensorFlow: TensorFlow is an open-source machine learning framework used for building and training deep learning models. The code imports TensorFlow to leverage its functionalities for constructing and training the ResNet50-based model.

OpenCV (cv2): OpenCV is a computer vision library used for image and video processing. It is employed in the code for reading video frames, resizing images, and annotating frames with detection labels.

NumPy: Fundamental package for scientific computing with Python.

ResNet50: The ResNet50 model is a pre-trained convolutional neural network (CNN) that has proven effective in image classification tasks. The code utilizes the ResNet50 model from Keras applications, with weights pre-trained on the ImageNet dataset.

Image Data Generator: The ImageDataGenerator class from Keras is employed for data augmentation during training. It applies transformations such as rescaling, shearing, zooming, and horizontal flipping to diversify the training dataset.

Model Construction and Compilation: The ResNet50-based model is defined and compiled using Keras. It includes a global average pooling layer, a fully connected layer with 512 units and ReLU activation, a dropout layer, and a final dense layer with softmax activation.

Frame Capture and Detection Functions: The code includes functions for detecting activities in video frames and saving annotated frames based on the trained ResNet50 model. The detect_activity function predicts the activity label and confidence for a given frame, and capture_frames processes a video, detecting and saving frames with specified confidence levels.

3.2 Literature survey

Title	Objective	Contribution	Methodology	Conclusion
An Efficient Violence Detection System from Video Clips using ConvLSTM and Keyframe Extraction.	Propose a novel violence detection model with improved accuracy and speed through clustering-based keyframe extraction and ConvLSTM network, addressing manual limitations and eliminating inefficiencies from unnecessary frames.	This research contributes significantly to urban violence detection by introducing the ECLVDS model, utilizing clustering-based keyframe extraction for efficiency, and enhancing spatiotemporal relations through a ConvLSTM network.	Proposed model employs clustering-based keyframe extraction, VGG-19 for spatial features, and ConvLSTM for classification, rigorously evaluated on the Hockey Fight Dataset for effective violence detection.	In conclusion, the ECLVDS model is a robust solution for automated urban violence detection, enhancing accuracy through clustering-based keyframe extraction.
Efficient Violence Detection Using 3D Convolutional Neural Networks	Developing an efficient model for automated analysis of violent content in surveillance videos using 3D CNNs, without relying on hand-crafted features or exclusive RNN architectures, for applications in Internet video filtration and public security protection.	Developing an effective model for automated analysis of violent content in surveillance videos using 3D CNNs, without hand-crafted features or exclusive RNN architectures, with applications in Internet video filtration and public security protection.	The model is trained and evaluated on three benchmark datasets: Hockey Fights (1000 videos, 720x576 frames), Movies (200 videos), and Violent-Flows (246 videos, 320x240 pixels).	Tests on regular benchmark datasets and extra experiments on the Mix dataset show that this model is better than existing methods in accurately recognizing violent actions.

Violence Detection Algorithm Based on Local Spatio-temporal Features and Optical Flow	Developing an algorithm for violence detection using local spatio-temporal features and optical flow, providing a novel data representation for enhanced capabilities in monitoring systems irrespective of contextual variations and the number of individuals involved.	Proposing an accuracy-enhancing violence detection approach with a physical contact detection algorithm. The Harris 3D detector and optical flow vector combination addresses computational challenges for a practical and efficient solution.	The research uses a smart physical contact detection algorithm to identify potential violence in videos. By combining the Harris 3D detector and optical flow, it overcomes computational challenges.	By combining a physical contact detection algorithm with the Harris 3D detector and optical flow, the approach effectively addresses challenges in standard methods, providing a more efficient way to identify potential violence.
---	---	--	---	---

3.3 Block diagram

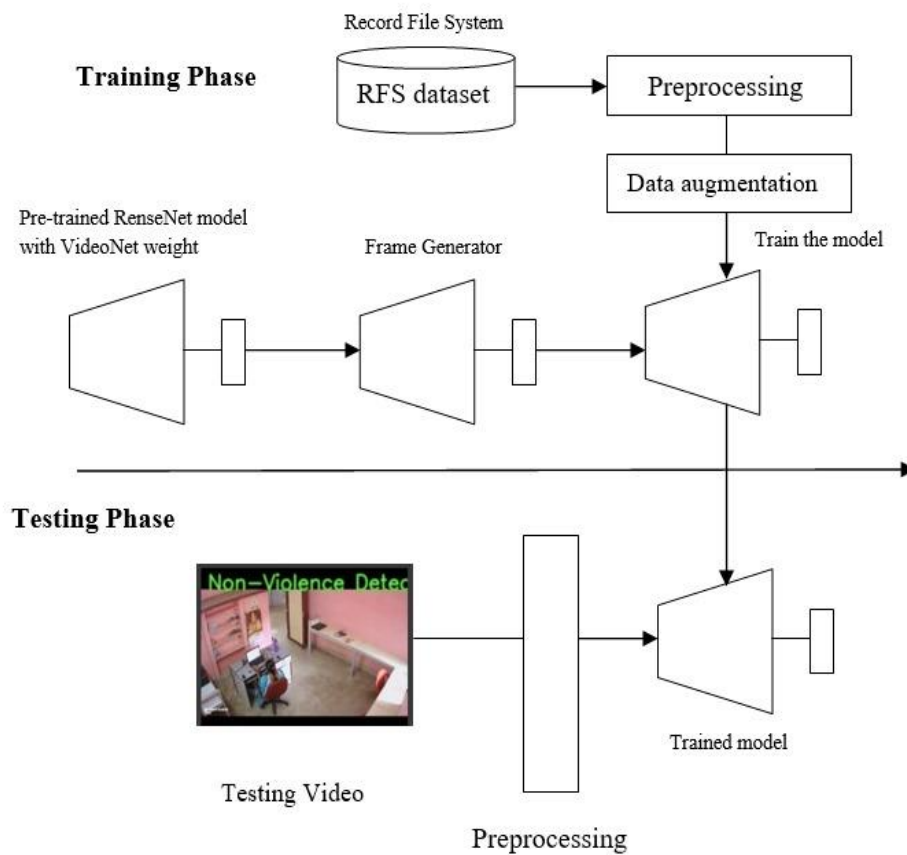


Fig 3.3 Block Diagram

- ❖ **Input:** The system receives a video feed or a sequence of frames from a video source, each frame capturing a scene snapshot.
- ❖ **Frame Preprocessing:** Frames undergo preprocessing, ensuring uniformity and compatibility with the ResNet50 model, commonly involving resizing for consistent dimensions.
- ❖ **ResNet50:** The pre-trained ResNet50 model is a crucial element, serving as a feature extractor to learn hierarchical representations from input frames. ResNet50's architecture facilitates feature reuse and diversity, making it effective for image classification tasks.
- ❖ **Softmax Activation:** The final layer utilizes a softmax activation function, converting model output into probabilities, assigning a probability distribution to classes for violent or non-violent categories.
- ❖ **Decision Making:** The model's output is analyzed, making decisions on violence and non-violence detection. A set threshold determines the class label based on the highest predicted probability, classifying frames as violent or non-violent accordingly.
- ❖ **Results:** The system presents final results, indicating the classification of each frame as violent or non-violent using the ResNet50 model.

3.4 Usecase Diagram

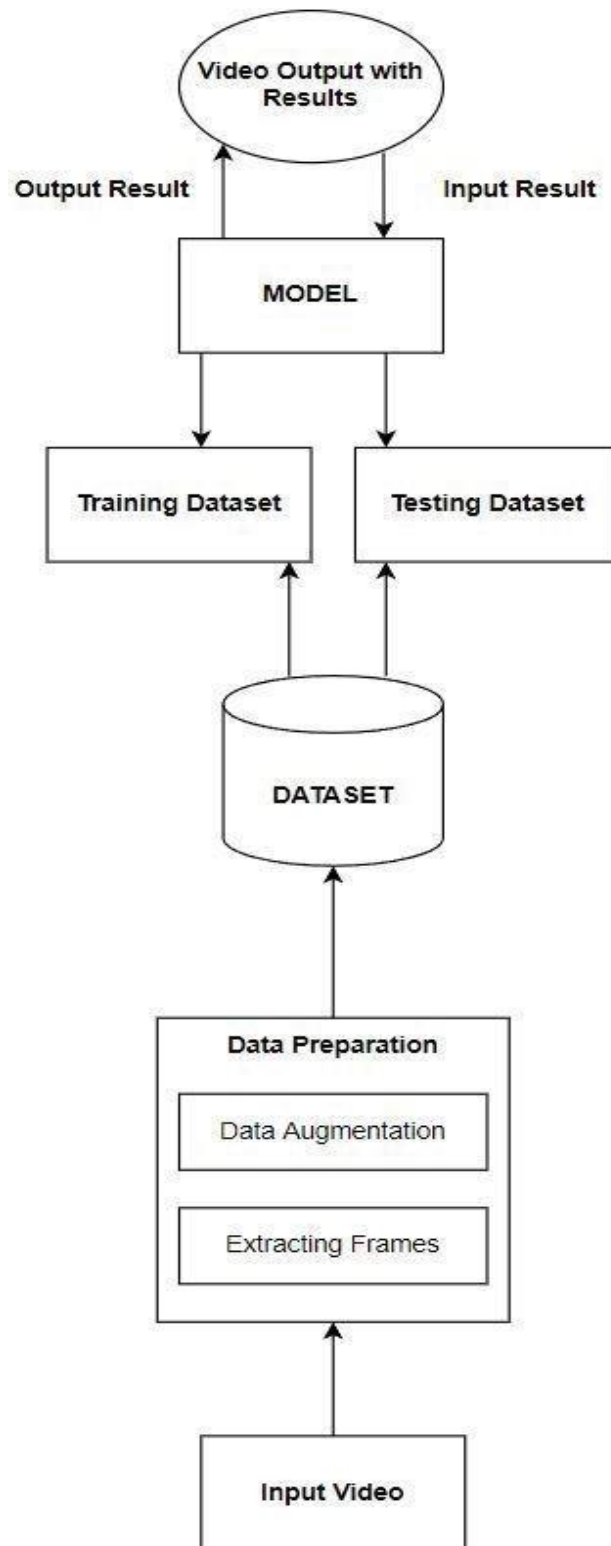


Fig 3.4: Use case diagram

The use case diagram for the ResNet50 model involves several key components. The "Input Video" serves as the initial interaction point, providing the video feed or sequence of frames for analysis. The "Data Preparation" step encompasses processes like Data Augmentation and

Extracting Frames, ensuring the input is suitably formatted for the ResNet50 model. The "DATASET" section includes both the Training Dataset and Testing Dataset, crucial for training and evaluating the model's performance. The "MODEL" component involves the Input Result, representing the data fed into the ResNet50 model, and the Output Result, depicting the model's predictions. Lastly, the "Video Output with Results" demonstrates the final presentation of the analyzed video, indicating whether each frame is classified as violent or non-violent based on the ResNet50 model's predictions.

4. RESULTS

4.1 SOFTWARE DESCRIPTION

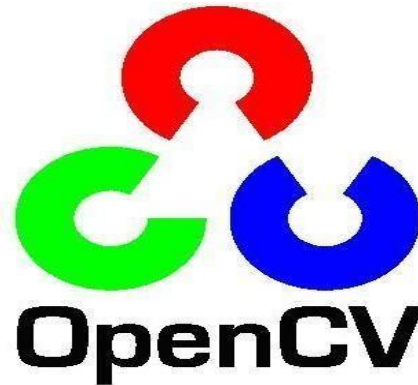


Fig4.1 OpenCV

OpenCV, or Open-Source Computer Vision Library, is an open-access computer vision and machine learning software library. Developed to provide a comprehensive set of tools for image and video processing, OpenCV is widely used in various fields, including robotics, augmented reality, and computer vision applications. It supports multiple programming languages, with bindings available for C++, Python, and Java, among others. Key features of OpenCV include image processing algorithms, computer vision functions, and machine learning tools. It encompasses a broad spectrum of tasks, such as image manipulation, object detection, facial recognition, and optical character recognition (OCR). Overall, OpenCV plays a pivotal role in advancing computer vision applications and research by providing a versatile and efficient framework for image and video analysis.



Fig4.2.2 Tensor Flow

TensorFlow is an open-source machine learning framework developed by the Google

Brain team. Renowned for its versatility, TensorFlow facilitates the creation and deployment of machine learning models across various platforms. Its flexible architecture supports both deep learning and traditional ML techniques, empowering developers to design and train complex neural networks. TensorFlow's high-level APIs simplify model development, while its low-level APIs offer granular control for advanced users. With support for multiple programming languages, including Python and C++, TensorFlow is widely adopted in diverse fields such as computer vision, natural language processing, and reinforcement learning. Its scalable nature and integration with hardware accelerators contribute to its popularity in both research and production environments.

4.2. Results

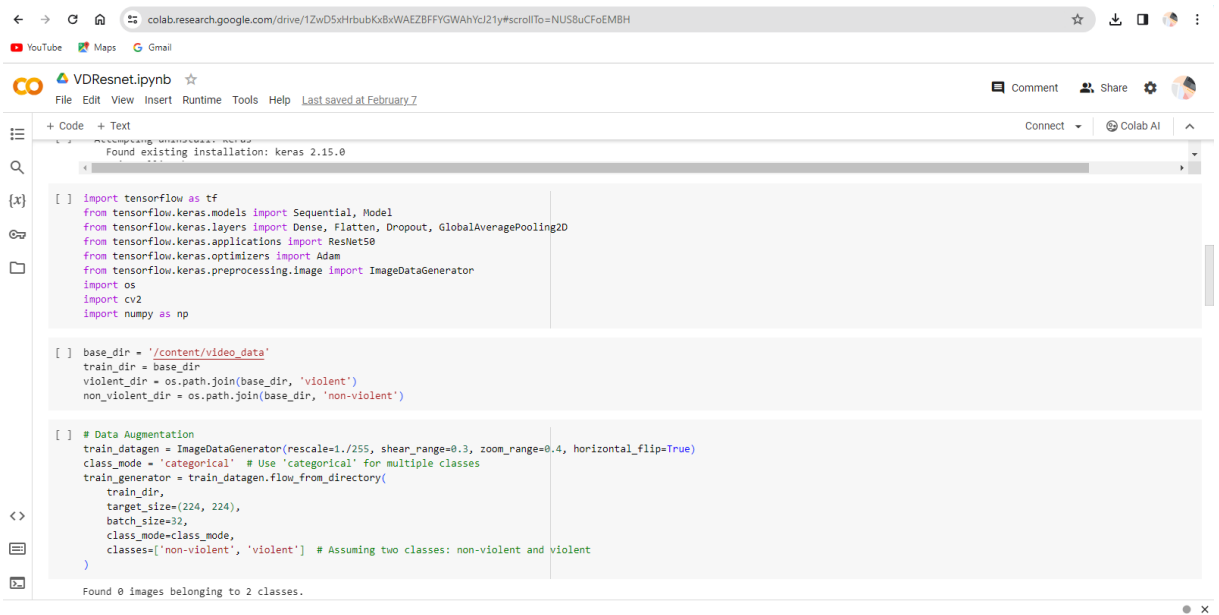
```

← → ↺ colab.research.google.com/drive/1ZwD5xHrbubKxWAEZBFYGVWAhYc21y
VDResnet.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
RAM Disk Colab AI
!pip install tensorflow==2.14.0
!pip install opencv-python

Collecting tensorflow==2.14.0
  Downloading tensorflow-2.14.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (489.8 MB)
    489.8/489.8 MB 2.4 MB/s eta 0:00:00
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.14.0) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.14.0) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.14.0) (23.5.26)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.14.0) (0.5.4)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.14.0) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.14.0) (3.9.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.14.0) (16.0.6)
Requirement already satisfied: ml-dtypes>=0.2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.14.0) (0.2.0)
Requirement already satisfied: numpy>=1.23.5 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.14.0) (1.23.5)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.14.0) (3.3.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.14.0) (23.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.14.0) (67.7.2)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.14.0) (67.7.2)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.14.0) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.14.0) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.14.0) (4.9.0)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.14.0) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.14.0) (0.35.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.14.0) (1.60.1)
Collecting tensorboard<2.15,>=2.14 (from tensorflow==2.14.0)
  Downloading tensorboard-2.14.1-py3-none-any.whl (5.5 MB)
    5.5/5.5 MB 77.6 MB/s eta 0:00:00
Collecting tensorflow-estimator<2.15,>=2.14.0 (from tensorflow==2.14.0)
  Downloading tensorflow_estimator-2.14.0-py2.py3-none-any.whl (440 kB)
    440.7/440.7 kB 34.2 MB/s eta 0:00:00
Collecting keras<2.15,>=2.14.0 (from tensorflow==2.14.0)
  Downloading keras-2.14.0-py3-none-any.whl (1.7 MB)
    1.7/1.7 MB 65.4 MB/s eta 0:00:00
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow==2.14.0) (0.42.0)
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.14.0) (2.17.3)
Collecting google-auth-oauthlib<1.1,>=0.5 (from tensorflow==2.14.0)
  Downloading google_auth_oauthlib-1.0.0-py2.py3-none-any.whl (18 kB)
    18/18 kB 12.5 MB/s eta 0:00:00
✓ 1m 12s completed at 8:27 PM

```

Fig. 4.2.1 : Installing tensorflow and OpenCV libraries



The image shows a Google Colab notebook titled "VDResnet.ipynb". The code is as follows:

```
Found existing installation: keras 2.15.0

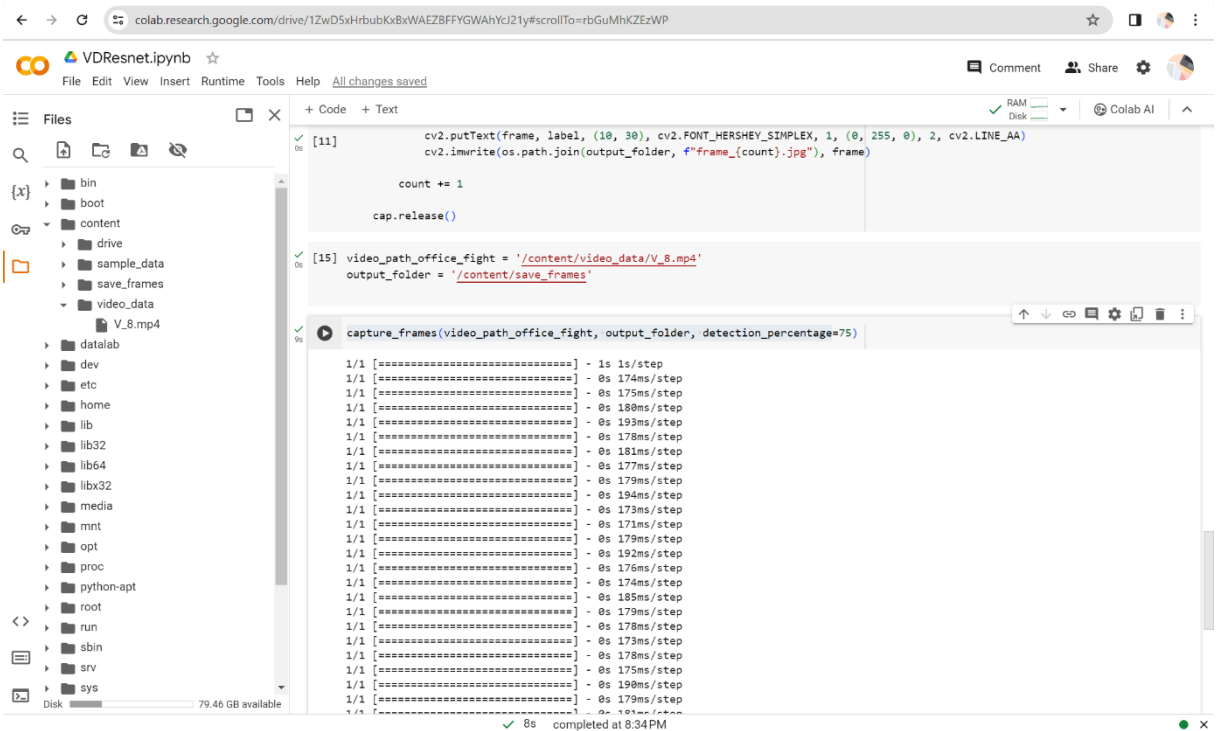
[ ] import tensorflow as tf
    from tensorflow.keras.models import Sequential, Model
    from tensorflow.keras.layers import Dense, Flatten, Dropout, GlobalAveragePooling2D
    from tensorflow.keras.applications import ResNet50
    from tensorflow.keras.optimizers import Adam
    from tensorflow.keras.preprocessing.image import ImageDataGenerator
    import os
    import cv2
    import numpy as np

[ ] base_dir = '/content/video_data'
    train_dir = base_dir
    violent_dir = os.path.join(base_dir, 'violent')
    non_violent_dir = os.path.join(base_dir, 'non-violent')

[ ] # Data Augmentation
    train_datagen = ImageDataGenerator(rescale=1./255, shear_range=0.3, zoom_range=0.4, horizontal_flip=True)
    class_mode = 'categorical' # Use 'categorical' for multiple classes
    train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(224, 224),
        batch_size=32,
        class_mode=class_mode,
        classes=['non-violent', 'violent'] # Assuming two classes: non-violent and violent
    )

Found 0 images belonging to 2 classes.
```

Fig. 4.2.2 : Importing tensorflow libraries



The image shows the same Colab notebook at a later stage. The left sidebar shows a file explorer with the following structure:

- bin
- boot
- content
 - drive
 - sample_data
 - save_frames
 - video_data
 - V_8.mp4
- datalab
- dev
- etc
- home
- lib
- lib32
- lib64
- libx32
- media
- mnt
- opt
- proc
- python-apt
- root
- run
- sbin
- srv
- sys

The main code area shows the following code cells:

```
[11] cv2.putText(frame, label, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2, cv2.LINE_AA)
      cv2.imwrite(os.path.join(output_folder, f"frame_{count}.jpg"), frame)

      count += 1

      cap.release()

[15] video_path_office_fight = '/content/video_data/V_8.mp4'
      output_folder = '/content/save_frames'

capture_frames(video_path_office_fight, output_folder, detection_percentage=75)
```

The output of the `capture_frames` function is displayed as follows:

```
1/1 [=====] - 1s 1s/step
1/1 [=====] - 0s 174ms/step
1/1 [=====] - 0s 175ms/step
1/1 [=====] - 0s 180ms/step
1/1 [=====] - 0s 193ms/step
1/1 [=====] - 0s 178ms/step
1/1 [=====] - 0s 181ms/step
1/1 [=====] - 0s 177ms/step
1/1 [=====] - 0s 179ms/step
1/1 [=====] - 0s 194ms/step
1/1 [=====] - 0s 173ms/step
1/1 [=====] - 0s 171ms/step
1/1 [=====] - 0s 179ms/step
1/1 [=====] - 0s 192ms/step
1/1 [=====] - 0s 176ms/step
1/1 [=====] - 0s 174ms/step
1/1 [=====] - 0s 185ms/step
1/1 [=====] - 0s 179ms/step
1/1 [=====] - 0s 178ms/step
1/1 [=====] - 0s 173ms/step
1/1 [=====] - 0s 178ms/step
1/1 [=====] - 0s 175ms/step
1/1 [=====] - 0s 190ms/step
1/1 [=====] - 0s 179ms/step
1/1 [=====] - 0s 181ms/step
```

At the bottom, it says "8s completed at 8:34 PM".

Fig. 4.2.3: Video framing

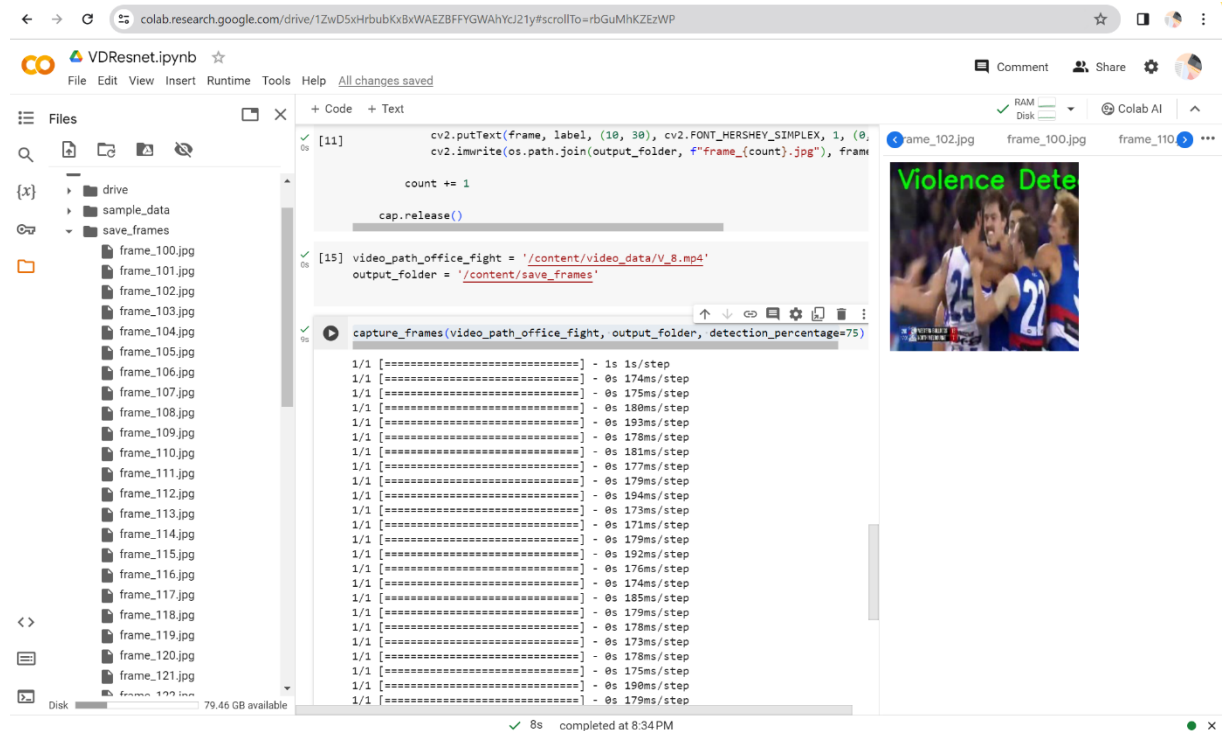


Fig. 4.2.4: Violence detection

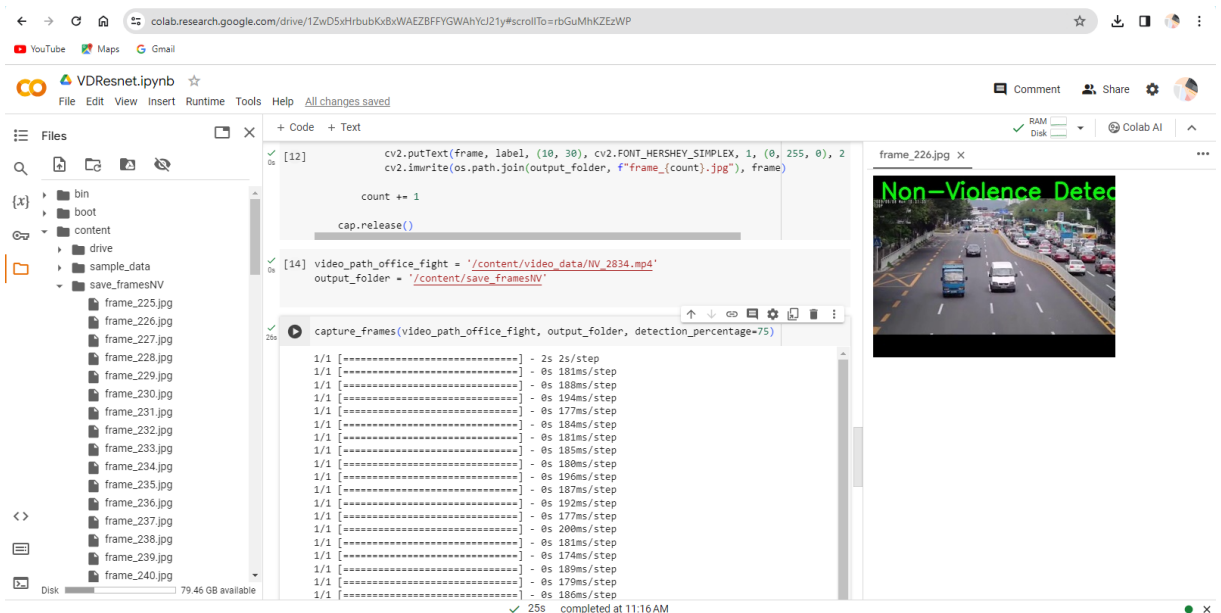


Fig. 4.2.5: Non-violence detection

CONCLUSION

The internship provided valuable insights into the practical applications of deep learning in public safety. The successful development of a real-time violence detection system using a ResNet50-based video analytics framework demonstrated the effectiveness of the approach. The challenges encountered highlighted the importance of addressing data quality issues and ethical considerations in developing AI solutions for sensitive domains. Overall, the internship significantly contributed to the enhancement of both scientific and professional skills, preparing for future endeavors in the field of artificial intelligence and public safety.

REFERENCES

- Nguyen, H. H., Ta, T. N., Nguyen, N. C., Pham, H. M., & Nguyen, D. M. (2021, January). Yolo based real-time human detection for smart video surveillance at the edge. In 2020 IEEE Eighth International Conference on Communications and Electronics(ICCE) (pp. 439-444). IEEE.
- Xu, Z., Li, J., & Zhang, M. (2021). A surveillance video real-time analysis system based on edge-cloud and fl-yolo cooperation in coal mine. IEEE Access, 9, 68482-68497.
- Goyal, S., Shagill, M. D., Kaur, A., Vohra, H., & Singh, A. (2020). A yolo based technique for early forest fire detection. Int. J. Innov. Technol. Explor. Eng.(IJITEE) Vol, 9, 1357-1362.