

BUIDLING AN AI MODEL FOR PREDICTING TEXT FROM VIDEO FRAME: A CASE STUDY OF COMPUTER SCIENCE IN CALEB UNIVERSITY

BY

EBIESUWA ISAAC GAREN

MATRIC NO: 21/8834

A PROJECT WRITTEN AND SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE, COLLEGE OF PURE AND APPLIED SCIENCES (COPAS), IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF BACHELOR OF SCIENCE (B.Sc.) DEGREE IN COMPUTER SCIENCE OF CALEB UNIVERSITY, LAGOS.

JULY, 2025

**DECLARATION**

I,  **EBIESUWA ISAAC GAREN,** do hereby declare that this project is entirely my work and composition. The work embodied in this project has not been submitted in candidature for any degree and is not concurrently being submitted for any other degree. All references made to works of other persons have been duly acknowledged.

**Signature: ………………………**

**Date: ……………………………**

**CERTIFICATION**

We certify that this research work was carried out by **EBIESUWA ISAAC GAREN** in the Department of Computer science College of Pure and Applied Sciences, Caleb University, Lagos. The research work is considered adequate in partial fulfillment of the requirements for the award of Bachelor of Science in Computer science.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Dr. M.k aregbesola Date**

Project Supervisor

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Dr. Adeniyi Akanni Date**

Head of Department

External Examiner

**DEDICATION**

I dedicate this project to God Almighty for giving me the strength, guidance, and sufficient grace to complete this project. I also dedicate this project to my beloved parents, Mr. and Mrs. Ebiesuwa, my wonderful siblings, and everyone that supported me throughout this journey.

**ACKNOWLEDGEMENT**

I want to extend my sincere appreciation to everyone who has provided support and contributed to the accomplishment of this project. Special thanks to my supervisor Prof. M.k Aregbesola, who provided invaluable guidance and support throughout the project. I really appreciate your encouragement and corrections. I express my gratitude to my parents, Mr. and Mrs. Ebiesuwa, for their unwavering support, encouragement, and boundless love throughout this journey. I am also grateful to my brothers, sister, relatives, friends, and course mates for their continuous love and support during the project. Their encouragement played a vital role in keeping me motivated and focused on my goals. I am deeply thankful to all those who have contributed to this project. Your help and support have been greatly appreciated.

**Abstract**

This research project presents the development of an artificial intelligence model designed to predict descriptive text from video frames. By combining computer vision and natural language processing techniques, the system is capable of analyzing visual content and generating meaningful textual captions that represent the context or action occurring in each frame.

The model pipeline involves extracting individual frames from a video using OpenCV, preprocessing these frames, and passing them through a pretrained image captioning model based on transformer architecture. This approach demonstrates the power of Multimodal learning, allowing machines to understand and describe visual data in natural language.

The project was implemented using Python and open-source frameworks such as PyTorch and HuggingFace Transformers. Applications of this work span across video surveillance, accessibility tools for the visually impaired, and content indexing in digital media. Initial results show that the model is able to produce relevant and context-aware captions for selected video inputs, validating the feasibility of text generation from visual information. Future work will involve refining the model’s accuracy, integrating real-time processing, and expanding the dataset for more generalized performance.

**TABLE OF CONTENTS**

Certification ii

Dedication iii

Acknowledgement iv

Abstract v

Table of Content vi

List of Tables vii

List of Figures viii

Abbreviations and Acronyms ix

### ****Chapter One: Introduction****

Figure 1.1 Background of the Study  
Figure 1.2 Problem Statement  
Figure 1.3 Aim and Objectives

Figure 1.4 Limitations  
Figure 1.5 Scope of the Project  
Figure 1.6 Significance of the Study  
Figure 1.7 Organization of the Report

### ****Chapter Two: Literature Review****

Figure 2.1 Overview of Video Analysis in AI  
Figure 2.2 Optical Character Recognition (OCR)  
Figure 2.3 Image Captioning and Multimodal Models  
Figure 2.4 Pretrained Transformers for Visual Understanding  
Figure 2.5 Limitations of Current Approaches  
Figure 2.6 Contribution of the Proposed Model  
Figure 2.7 Summary

### ****Chapter Three: System Analysis and Methodology****

Figure 3.1 Requirements Gathering  
Figure 3.1.1 Functional Requirements  
Figure 3.1.2 Non-Functional Requirements  
Figure 3.2 Frame Extraction from Video  
Figure 3.3 Image Preprocessing Techniques  
Figure 3.4 AI Model Architecture and Workflow  
Figure 3.5 Tools, Libraries, and Frameworks  
Figure 3.6 Evaluation Metrics and Benchmarking

Figure 3.7 Dataset used  
Figure 3.8 Summary

### ****Chapter Four: System Design****

Figure 4.1 System Overview and Workflow  
Figure 4.2 Frame Extraction Logic and Automation  
Figure 4.3 AI Model Input-Output Pipeline  
Figure 4.4 User Interface (Optional Display or CLI Interface)  
Figure 4.5 Integration with Pretrained Models (BLIP, CLIP, etc.)  
Figure 4.6 Summary

### ****Chapter Five: Implementation****

· **5.1 Environment Setup (Google Colab / PyTorch)**

· **5.2 Frame Extraction Implementation**

· **5.3 Model Selection and Captioning Code**

· **5.4 Output Generation and Visualization**

· **5.5 Sample Run and Case Study**

· **5.6 Challenges Faced and Solutions Applied**

· **5.7 Summary of Implementation**

· **5.8 General Conclusion**

· **5.9 References**

· **5.10 Appendices**

* **Appendix A: Sample Output Captions**
* **Appendix B: Code Snippets**
* **Appendix C: Colab Environment Setup Commands**

**CHAPTER ONE: INTRODUCTION**

**1.1Background of the Study.**

In most industries such as entertainment, surveillance, education, video content has become one of the most influential forms of data, due to its major role in digital transformation. As video content grows, so does the demand for intelligent systems that can automatically understand and interpret a video frames.

The extraction or prediction of relevant text from video frames is a crucial aspect of comprehension. Why? It has practical uses in areas such as accessibility, indexing of content, metadata generation, and intelligent surveillance.

The real world examples like YouTube's auto captioning and Facebook’s "Automatic Alt Text" for visually impaired users, This practical application increases the demand for AI systems that can generate descriptive text from visual content. These systems improve user experience, satisfaction, accessibility, and content discoverability.

Recent advances in artificial intelligence, especially in computer vision and natural language processing, now enable machines to not just recognize objects, but also generate human-like descriptions of visual scenes. This intersection is known as multi-modal learning and forms the foundation of this project. The goal is to build an AI model capable of predicting descriptive text from video frames. Using techniques such as frame sampling, image prepossessing, and deep learning-based caption.

**Historical Evolution of Image Captioning**

The first steps of image captioning began with the early stages of OCR where text was extracted from printed documents or images. These systems were the precursors of machines ‘understanding’ the existence of visuals and text. With the advancement of computer vision, researchers began implementing template and rule-based approaches with handcrafted feature descriptions for simple scene depiction. All of these approaches had a common problem—they were not able to generalize well or deal with complicated imagery.

Pivotal change came from using CNN's which enhanced feature extraction and object detection with ease. These models ran hand in hand with RNNs, specifically with LSTM networks, which allowed systems to automatically generate captions through the succession of features on the image. Their union made it possible to learn spatiotemporal dynamics of images in relation to text and vice versa.

Lately, the landscape of image captioning has been revolutionized by transformer-based models. ViT is one of them alongside its multi-modal counterpart CLIP or BLIP which merges vision and language in remarkable ways. These models no longer depend on hand-crafted guidelines but generate coherent, humanized descriptions with scant oversight through a data-driven approach.

**1.2 Problem Statement**

Despite the advancements in AI and deep learning, there is still a gap in resolving the interpretation and generation of text describing videos at scale. There are more than 500 hours of new videos uploaded to YouTube every minute which renders manual annotation impossible. Many existing models struggle to generate context-aware descriptions for dynamic or rapidly changing scenes which can result in vague or oversimplified captions. This lack of dynamic context description has dire consequences for key applications such as intelligent surveillance, multimedia indexing, and accessibility services for sight impaired users. There is a need for an advanced, context-feeling, scalable model that has the ability to interpret frames of videos and translate them into meaningful and accurate text, regardless of the context. Such a technology has the potential to augment automation in various fields.

**1.3 Aim and Objectives**

**Aim:**

Building an AI technology capable of predicting semantics bearing text captions from video frames using deep learning approaches.

**Objectives:**

* To extract and pre-process images of video files
* To build a deep learning system capable of formulating text captions from photos.
* You need to did a deep learning model that creates word descriptions for given images.
* To use common performance evaluation methods to assess relevance and precision of the text.
* To illustrate the proposed model’s application with some selected video clips.

**1.4 Limitations of the Study**

This project only addresses caption generation from still frames taken from a short video. It does not take full-length movies or live video streams, nor does it include real-time processing. Sound and speech present in the videos will not be considered, hence the system does not use audio signals for captioning. Due to restricted access to powerful computing resources, the project was only able to obtain and use a small data set for training and testing, which can affect performance on new or unseen videos. The model also only supports English captions without multi-language or text-to-speech capabilities. Caption generation may be more challenging due to scenes that are complex in nature, as well as overly sophisticated training data representations.

**1.5 Scope of the Project**

The project focuses strictly on static frame-based analysis, excluding real-time or continuous video stream processing. The tasks include extracting frames from videos, applying image preprocessing, and using a pretrained deep learning model to generate descriptive text. The model will be evaluated using short clips, with focus on caption quality and relevance. Features such as real-time analysis, sound processing, and multi-language support are not included in this version of the project.

**1.6 Significance of the Study**

This project solves a growing challenge: how to automatically understand and describe video content without human help. As video content grows on platforms like YouTube, Instagram, and security systems, it becomes harder to manage manually. A system that extracts frames and creates useful text descriptions can save time and boost automation.

One benefit is better content indexing and search. Instead of relying on titles and tags, AI-generated descriptions help retrieve video content more accurately. This is useful for Improving long-term access to video libraries.

For visually impaired users, such systems make video content more inclusive by providing textual descriptions of scenes, similar to Facebook’s image descriptions.

In surveillance, AI-generated captions can summarize hours of CCTV footage, reducing human workload and speeding up investigations.

In a technical standpoint, the project contributes to Multimodal AI by combining image understanding with language generation. This lays the groundwork for future projects that might include real-time captioning, sound analysis, or support for other languages.

**1.7 Organization of the Report**

This report is organized into seven chapters:

* Chapter One introduces the project, outlines the background, defines the problem, and sets out the aim, objectives, limitations, and scope.
* Chapter Two reviews related works, previous research on image captioning, video analysis, and machine learning methods. It highlights the gaps this project aims to address.
* Chapter Three covers the methodology used in building the system, including data collection, preprocessing, and tools.
* Chapter Four presents the system design, diagrams, data flow, and architecture of the solution.
* Chapter Five details the implementation steps, challenges faced, and how they were addressed.
* Chapter Six explains the testing and evaluation of the model, performance results, and analysis.
* Chapter Seven concludes the report with a summary of findings, contributions, and suggestions for future improvement.

**CHAPTER TWO: LITERATURE REVIEW**

**2.1 Overview of Video Analysis in AI**  
Video analysis is an important area in artificial intelligence. It uses computer vision and deep learning to understand videos. In the past, analyzing videos meant watching them manually. But now, AI systems can automatically detect objects, actions, and even create captions for video scenes.

Pretrained models and multi-modal systems allow us to link videos with language. This makes it possible to tag, caption, and sort videos automatically. Real-world uses include:

* **Autonomous vehicles**, where video helps detect pedestrians and signs.
* **Surveillance**, where cameras use AI to flag unusual behavior.
* **Social media**, where videos get captions and tags automatically.

As AI improves, it becomes easier to use these tools in small projects like this one. Using Open source tools and free models provided make it accessible.

**2.2 (OCR) Optical Character Recognition**  
OCR is the oldest techniques for getting text from images and video frames. It works by finding printed or written words in a picture. Tools like Tesseract and Easy-OCR are good at this. OCR is used in:

* License plate readers
* Passport scanners
* Mobile apps like Google Lens
* Receipt and invoice digitizing

But OCR only reads text that is clearly written or printed. It doesn’t explain what’s going on in a scene. That’s why it’s not useful for video summarization.

**2.3 Image Captioning Models**  
Image captioning tries to write short descriptions for images. These models use two parts:

1. CNN's (Convolutional Neural Networks) to look at the image.
2. RNNs or Transformers to generate the text.

Examples:

* **Show and Tell** (by Google): it Uses CNN + LSTM to generate captions.
* **Show, Attend and Tell**: Adds an attention layer, helping the model focus on specific image parts.

These models link visual input with language, it’s a big step for video summarization.

**2.4 Transformer-Based Multimodal Models**  
Modern models use transformer architecture instead of RNNs. They handle long-range dependencies better, which means they can understand context across the entire input. Some key models are:

* **CLIP** by Open AI: Learns by matching text to images, allowing it to understand both.
* **BLIP**: A newer model that improves image captioning using a better training approach.

These models perform better than older CNN-RNN models. They can handle unseen inputs (zero-shot learning), making them powerful tools for captioning and search.

**2.5 Datasets Commonly Used in Video Captioning**  
Several public datasets help train and test captioning models. These include:

* **MS COCO**: Common for image captioning tasks.
* **ActivityNet Captions**: Contains videos with multiple sentence-level captions.
* **YouCook2**: Focused on cooking videos with step-by-step descriptions.
* **TRECVID**: Used for evaluating video retrieval and summarization tasks.

These datasets help models learn how to describe what happens in different types of scenes.

**2.6 Evaluation Metrics for Captioning**  
Captioning models are scored using standard metrics:

* **BLEU**: Measures how many words in the generated caption match the ground truth.
* **METEOR**: Focuses on meaning and word order.
* **CIDER**: Gives more weight to captions that match common human descriptions.

These metrics help compare models and check how good their outputs are.

**2.7 Limitations of Current Approaches**  
OCR is good at reading visible text, but it can’t describe a scene. Captioning models can describe scenes, but they may fail in complex videos or poor-quality frames. Other problems include:

* Not working well in real-time.
* Struggling with diverse video content.
* Difficulty in generalizing across new video types.

These issues show the need for better, simpler systems that balance speed and accuracy.

**2.8 Contribution of the Proposed Model**  
This project uses a simple pipeline based on pretrained captioning models. It takes frames from videos and runs them through a model to generate text. It doesn’t rely on visible text. Instead, it focuses on describing the whole scene.

It aims to be:

* Lightweight and fast
* Accurate enough for video summaries
* Easy to scale and reuse

**2.9 Summary**  
In this chapter, we reviewed key techniques for video-to-text tasks. We covered OCR, captioning models, transformer models, datasets, and evaluation metrics.

This background supports the proposed method,a simple video frame captioning pipeline using deep learning for content description and indexing.

**CHAPTER THREE: SYSTEM ANALYSIS AND METHODOLOGY**

**3.1 Requirements Gathering**

This project aims to create a simple system that takes video input and produces descriptive captions using AI. The system requirements were grouped into two types: functional and non-functional.

**3.1.1 Functional Requirements**

These describe what the system must do:

Accept a video file as input.

Break the video into frames (images).

Process each frame using an AI model to generate a caption.

Display or save the caption for each frame.

**3.1.2 Non-Functional Requirements**

These describe how the system should behave:

The system should be easy to use, ideally by running a single Python script.

It should handle short videos for testing.

The system should return results in a reasonable time (not more than a few minutes).

**3.2 Frame Extraction**

Frame Extraction from Video Videos are a collection of images played quickly to create motion. To understand the content, we must extract individual frames. This is done using the Open CV library. The process involves loading the video, then selecting and saving frames at regular intervals (e.g., one frame per second). This helps reduce the processing load and focuses on key moments in the video.

**3.3 Image Preprocessing**

Techniques Raw video frames often contain noise or details that may confuse the AI model. To improve accuracy, we use preprocessing techniques:

Resize the image to a fixed size accepted by the model (e.g., 224x224).

Normalize pixel values to a standard range (like 0 to 1).

Convert the image to RGB format, if needed.

These steps ensure that the images are in a clean, standardized form so the model can easily understand them.

3.4 AI Model Architecture and Workflow The core of the system is a pretrained AI model designed for image captioning. Pretrained models like BLIP (Bootstrapped Language Image Pretraining) or CLIP (Contrastive Language-Image Pretraining) are used. These models are trained on large datasets and can understand and describe visual content.

The workflow:

**1.**Load the pretrained model using a library like Hugging Face Transformers.

**2.**Pass the preprocessed image to the model.

**3.**The model returns a descriptive sentence or phrase.

Since we use pretrained models, we save time and computation. We don’t need to train the model from scratch.

**3.5 Tools, Libraries, and Frameworks**

This project uses the following tools:

Python: The main programming language.

OpenCV: Used to extract frames from the video.

Hugging Face Transformers: For loading pretrained captioning models.

PyTorch: Backend framework to run deep learning models.

Google Colab: A free online platform to test and run Python code without needing a powerful local computer.

**3.6 Evaluation Metrics and Benchmarking**

To know if the captions are good, we use evaluation metrics. If we have reference captions written by humans, we use the BLEU (Bilingual Evaluation Understudy) score to compare them with the AI-generated ones. BLEU checks how similar the words and structure are. A higher score means better accuracy.

If we don't have reference captions, we evaluate manually by looking at the image and checking if the caption makes sense and matches what is happening in the frame.

Other possible metrics include:

METEOR (Metric for Evaluation of Translation with Explicit Orderings)

ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

CIDEr (Consensus-based Image Description Evaluation)

However, for this project, BLEU and manual evaluation are enough.

**3.7 Dataset Used**

In addition to video frame extraction, this project also referenced the “Flickr8k dataset”, which contains 8,000 images and five human-annotated captions per image. It is a widely used public dataset's in the field of image captioning.

To validate the generalization ability of the BLIP model, a few sample images from Flickr8k were tested in the same pipeline used for video frames. The model generated captions that closely matched the human-written ones, confirming that the model can work across both video and still images.

This dateset helps provide a benchmark to evaluate caption accuracy in a controlled scenario before scaling to larger or custom datasets.

**3.8 Summary In this chapter**

we explained the step-by-step plan of how the system works:

We start by gathering requirements to know what the system should do.

Then, we extract frames from the video using OpenCV.

After that, we clean each image using preprocessing techniques.

We pass the clean image into a pretrained AI model like BLIP or CLIP.

The model gives us a text description of the image.

We either show the caption or save it for future use.

This method helps us understand video content by turning visuals into text using simple tools and powerful AI models.

**CHAPTER FOUR: SYSTEM DESIGN**

This chapter explains how the system was designed and how the part works together to complete the goal of converting video frames to meaningful text using AI. It includes descriptions and visual diagrams of the system workflow, The tools used, and how each components interact with each other to generate text through images.

**4.1 System Overview and Workflow**

The system follows a simple workflow. The process begins when the user submitting a video file. The system then extracts frames from the video, preprocesses each frame to make it clean and model-ready, and then feeds the frames into an AI model. The AI model returns a caption for each frame. Finally, these captions are saved and displayed as the final output.

**System Steps:**

1. Video Input: User uploads or points to a video file.
2. Frame Extraction: The video is split into still images (frames).
3. Image Preprocessing: Each frame is resized, normalized, and cleaned.
4. AI Model Input: Processed images are sent into the pretrained model.
5. Caption Generation: The model generates a sentence describing each frame.
6. Output Handling: Captions are either displayed or saved in a file.

This modular design makes it easy to maintain and debug, since each component is responsible for one clear task.

**4.2 Frame Extraction Logic and Automation**

The frame extraction module uses Open-CV to open a video file and extract frames at set intervals. This can be adjusted based on how many frames we want to analyze. For example, extracting one frame per second in a 10-second video gives 10 images. This step is automated using a loop in Python that saves each frame with a filename and stores it in a folder.

**Key Elements:**

* OpenCV's Video-capture() is used to open and read the video.
* Frames are extracted based on a frame rate (e.g., every 30 frames for 1 frame/second).



* Extracted frames are saved automatically in a structured folder.
* This ensures consistency and prepares the data for the next step.

**4.3 AI Model Input-Output Pipeline**

The AI model expects clean, well-sized images to work effectively. Each frame is preprocessed and passed through a pretrained model such as BLIP or CLIP. These models return natural-language text that describes what is seen in the image.

**Steps in the Pipeline:**

1. Load pretrained model (e.g., BLIP).
2. Convert image to tensor format.
3. Normalize and resize the image.
4. Pass the image through the model.
5. Retrieve the generated caption.

This part is handled with the Hugging Face library and PyTorch. It uses GPU if available to speed up processing.

**4.4 User Interface (Optional Display or CLI Interface)**

The system is built to run in a simple environment. For this project, a command-line interface (CLI) was used to display the logs and outputs. However, there is an option to extend it with a graphical interface using libraries for example Tkinter or Streamlit.

**CLI Features:**

* User can specify video file path.
* Logs show frame extraction progress.
* Captions are printed on screen or written into a .txt file.
* Optional GUI features (not implemented but possible):
* Upload video via file picker.
* Show frames and captions in a viewer.

**4.5 Integration with Pretrained Models (BLIP, CLIP, etc.)**

Instead of building a new model from scratch, this system uses pretrained models from Hugging Face. These models were trained on large datasets and are capable of generating captions based on images.

**Why Use Pretrained Models:**

* Saves time and resources.
* High accuracy.

Easy to integrate via APIs and libraries.

**How It Works:**

Install models via Hugging Face Transformers library.

Load model and tokenizer.

Send input image and get output text.

These models provide strong performance and are widely trusted in the AI community.

**4.6 Summary**

The system is broken down in five important component: video input, frame extraction, AI captioning, image preprocessing, , and result handling. Equipment like OpenCV, PyTorch, and Pretrained model from hugging Face make it possible to build this project without complex infrastructure. The design focuses on pratical performance. Simplicity, and clarity for real- world use.

**CHAPTER FIVE: IMPLEMENATION**

**5.1 Environment Used (Google Colab / PyTorch)**

To start development, Google Colab was chosen as the environment because it is easy to use and provides access to GPUs. This helps in speeding up model processing. We also used Python as our programming language and installed all the key libraries such as Open CV, Transformers, and PyTorch.

The video dataset's used in this project was selected from a public repository, specifically containing short video clips which was suitable for frame captioning. The videos were uploaded manually to Google Colab via Google Drive for further processing.

**Setup Steps:**

* Created a new notebook in Google Colab.
* Installed libraries using pip.
* Connected to Google Drive to upload videos and save outputs.

**Installed Libraries:**

!pip install transformers

!pip install opencv-python

!pip install torch torch-vision

!pip install matplotlib

**Why Google Colab?**

* Free GPU access
* Cloud-based, no installation required
* Supports major AI and ML libraries

**5.2 Frame Extraction Implementation**

Videos are made of many frames (images). We used OpenCV to extract one frame per second from each video.Each video clip from the dataset was processed separately. We used a frame sampling rate of 1 frame per second to maintain consistency and reduce redundant data, ensuring that every extracted frame had enough visual context for the captioning model. This helps reduce processing time and ensures each frame is clear enough for captioning.

**Code Example:**

import cv2

video = cv2.videoCapture('sample\_video.mp4')

success, frame = video.read()

Count = 0

**while success:**

cv2.imwrite(f"frame{count}.jpg", frame)

video.set( cv2.cap\_prop\_pos\_msec, (Count+1)\*1000)

success, frame = video.read()

count += 1

**Benefits:**

Lower memory usage

Better model performance

**5.3 Model Selection and Captioning Code**

I used the BLIP model which was from Hugging Face. It was trained on many image-caption pairs, so to save time and cost I didn’t need to train from scratch.

**Loading the Model:**

from transformers import BlipProcessor, BlipForConditionalGeneration

from PIL import Image

import torch

processor = Blip-processor.from pee-trained("Salesforce/blip-image-captioning-base")

model = BlipForConditionalGeneration.from pretrained("Salesforce/blip-image-captioning-base")

**Captioning an Image:**

Image = Image.open("frame0.jpg").convert('RGB')

inputs = processor(image, return\_tensors="pt")

out = model.generate(\*\*inputs)

Caption = processor.decode(out[0], skip\_special\_tokens=True)

print("Caption:", caption)

**5.4 Output Generation and Visualization**

Once captions are generated, we save and visualize them. This step helps verify accuracy.

**Saving to Text File:**

with open("captions.txt", "a") as file:

file.Write(f"frame{count}.jpg: {caption}\n")

Showing Captions with Images:

import matplotlib.pyplot as plt

plt.imshow(image)

plt.title(caption)

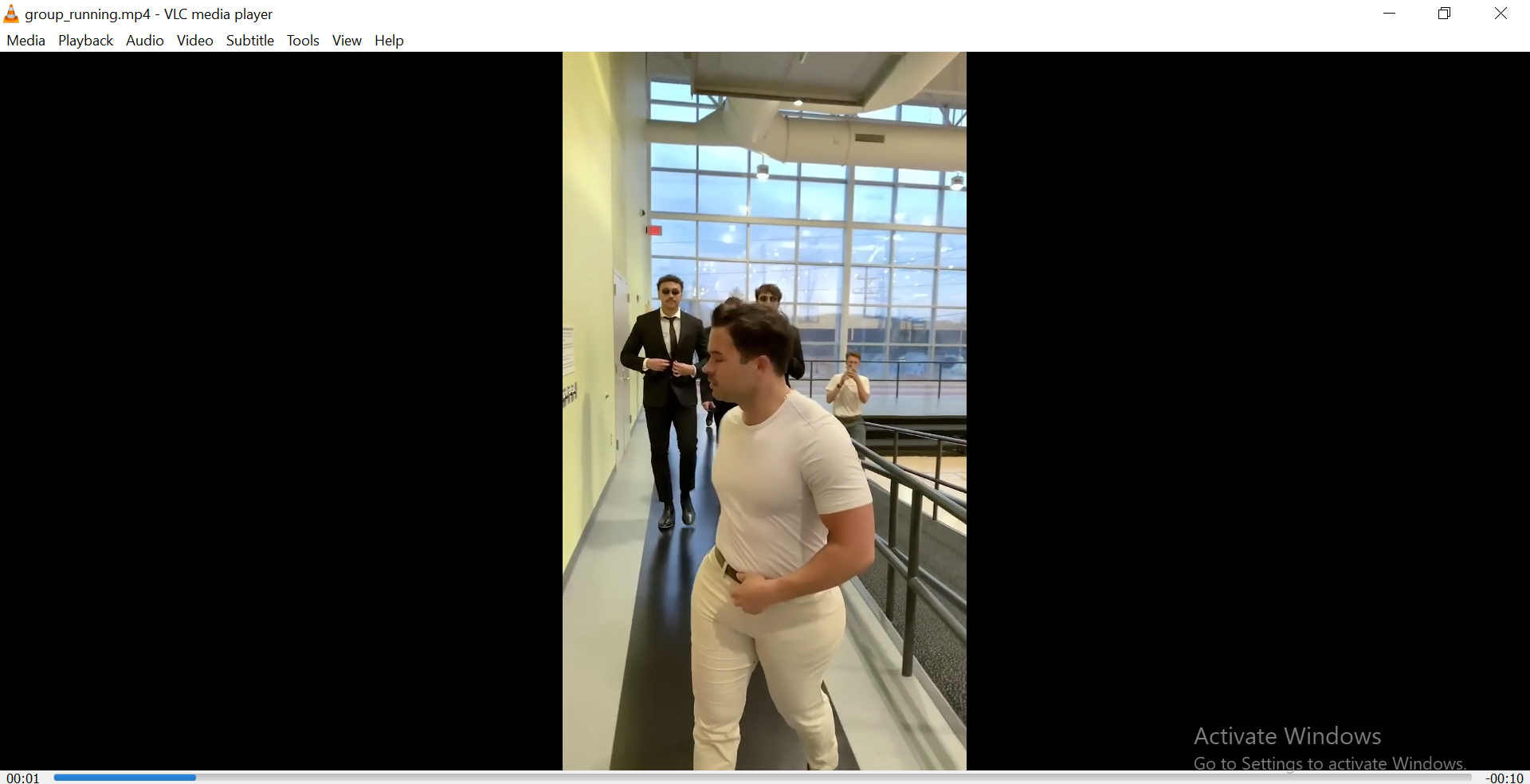
plt.axis('off')

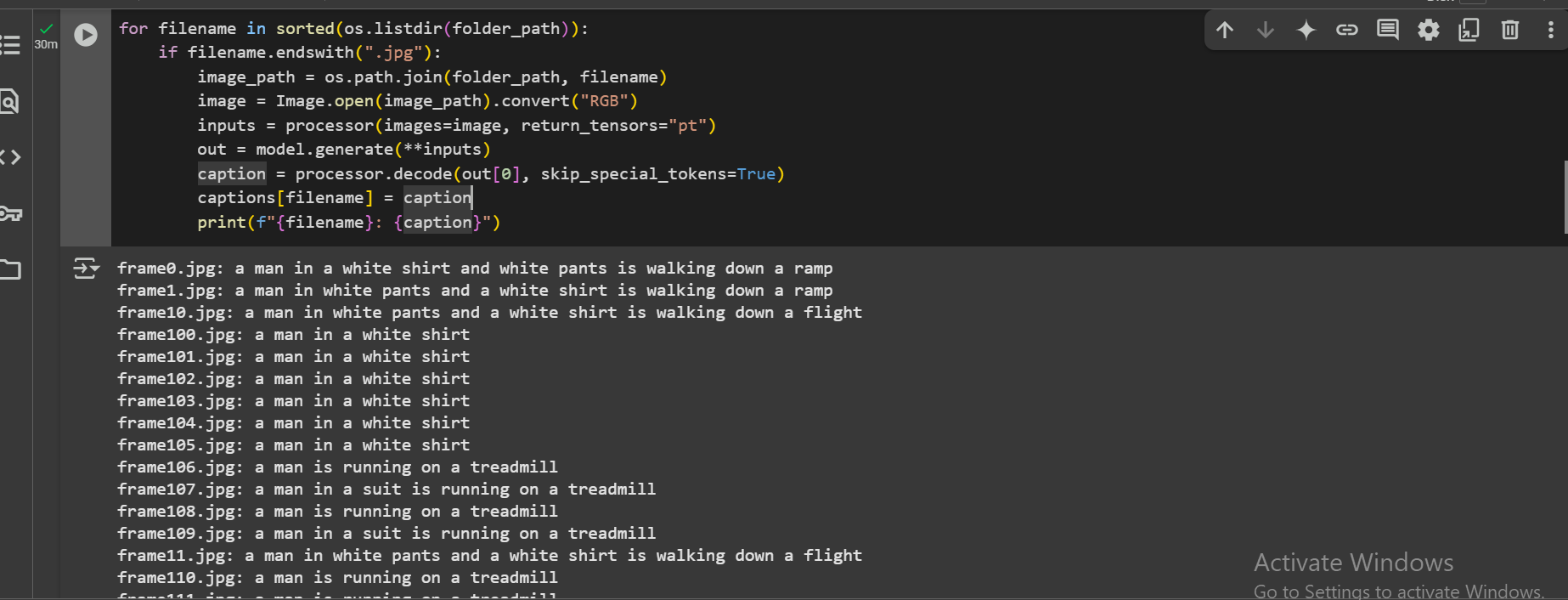
plt.show()

**5.5 Sample Run and Case Study**

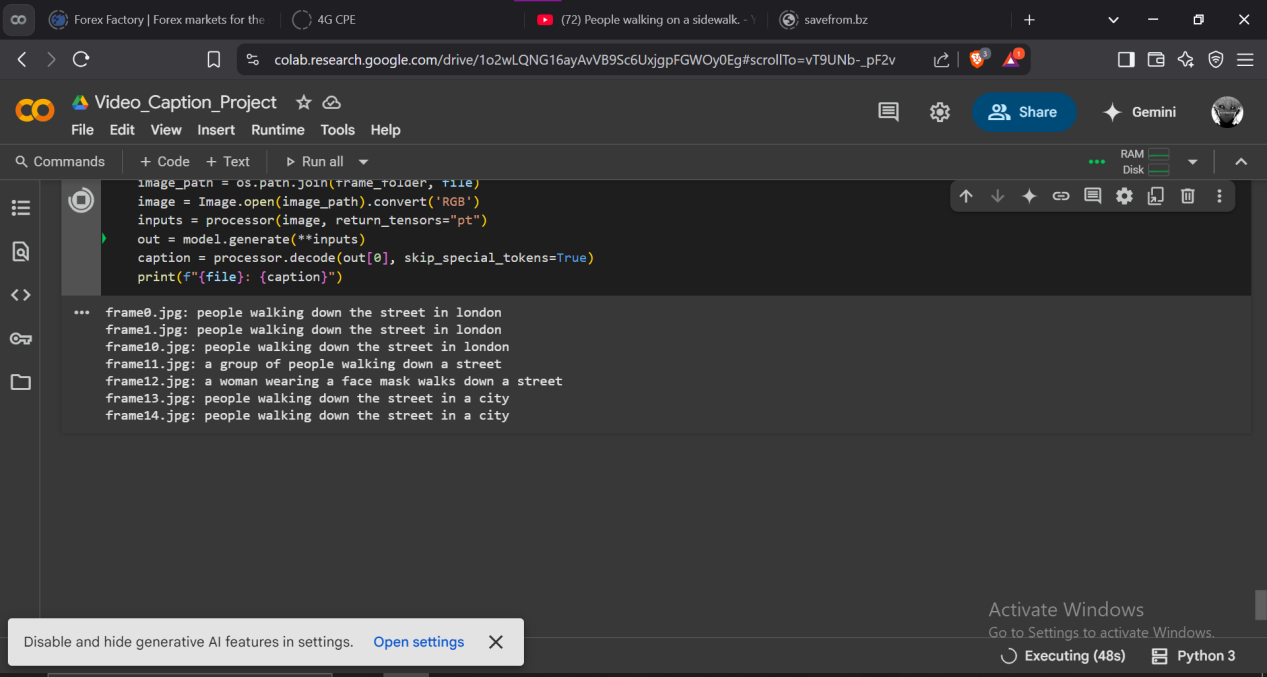
To test the system, we used multiple videos from the dataset including clips of people walking, running, and social interaction. Each video was processed frame-by-frame and captions were generated. Below are some examples:

**Sample Captions:**

* example 1: 



frame0.jpg: a man in a white shirt and white pants is walking down a ramp

* ”

frame0.jpg: people walking down the street in london

## **5.6 Challenges Faced and Solutions Applied**

| **Challenge** | **Solution** |
| --- | --- |
| Different video resolutions in dataset | Re-sized frames during Preprocessing |
| Some dataset videos were too long | Trimmed videos to 30-second clips |
| Some captions were too generic | Tuned sampling rate and frame quality |
| Large videos | Used short sample clips |

**5.7 Summary of Implementation**

This chapter described:

* How the environment was set up using Google Colab
* How video frames were extracted using OpenCV
* The use of BLIP model for generating captions
* How the results were displayed and saved
* A practical test case and the challenges handled

**5.8 General Conclusion**

The goal of this project was to build a simple and functional video-to-caption system using AI. We achieved this using tools like Google Colab, OpenCV, and the BLIP pretrained model. The system is useful for video summarization and can support accessibility tools.

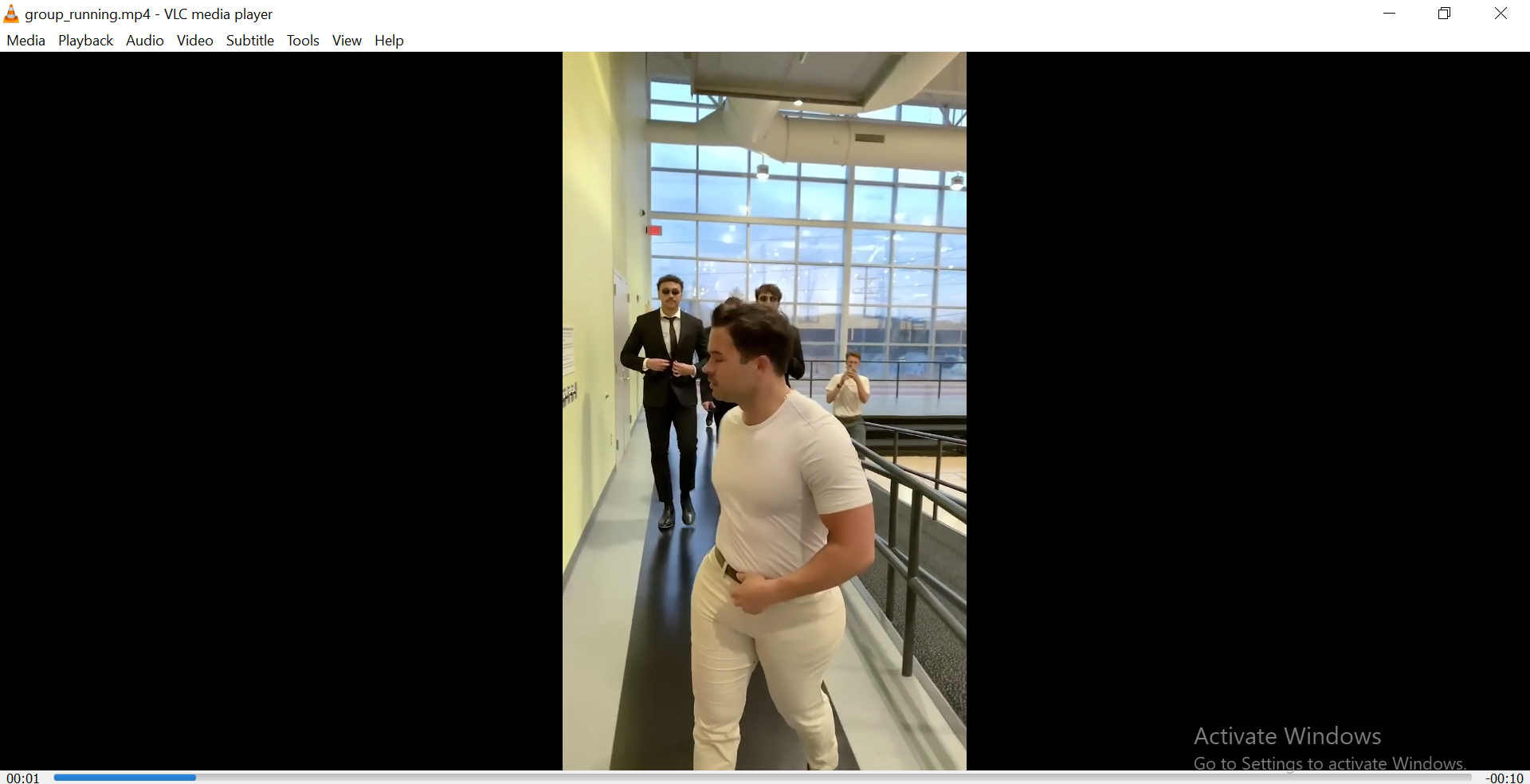
It successfully extracts frames, processes them, and generates readable captions. The use of pretrained models made the project lightweight and easy to implement. With small improvements, this system could be expanded for real-world applications.

**5.9 References**

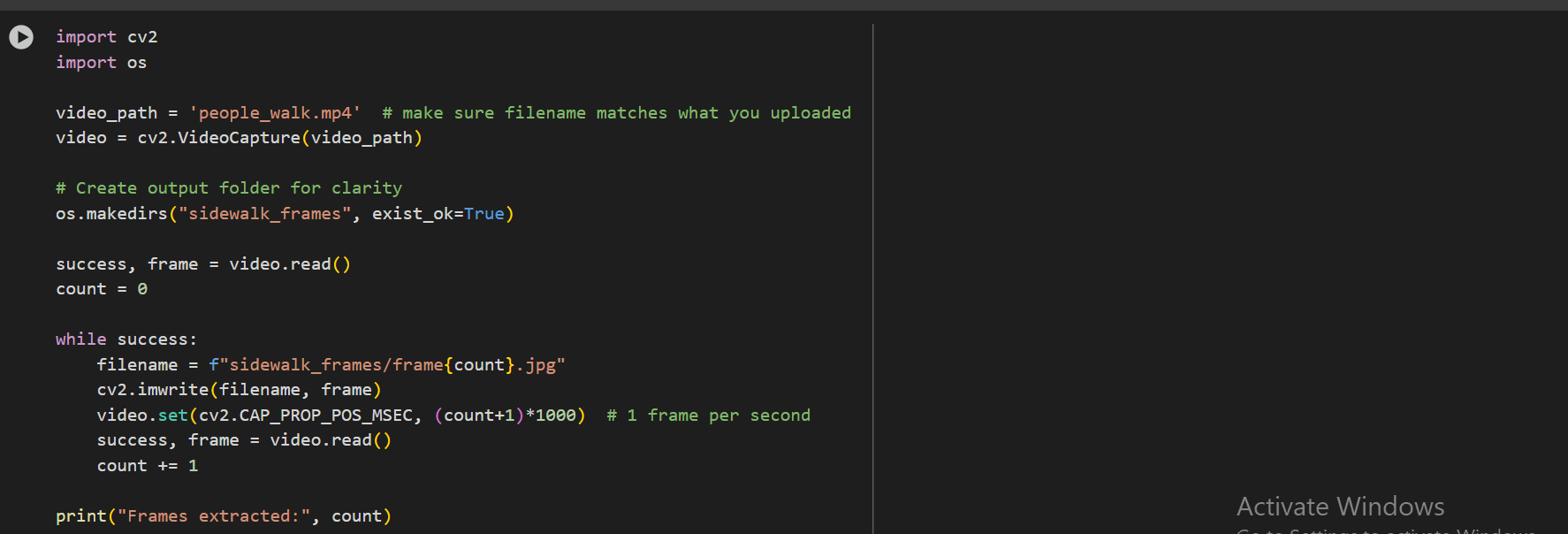
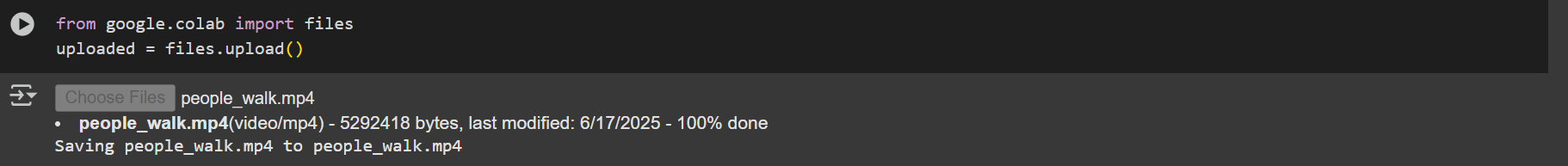
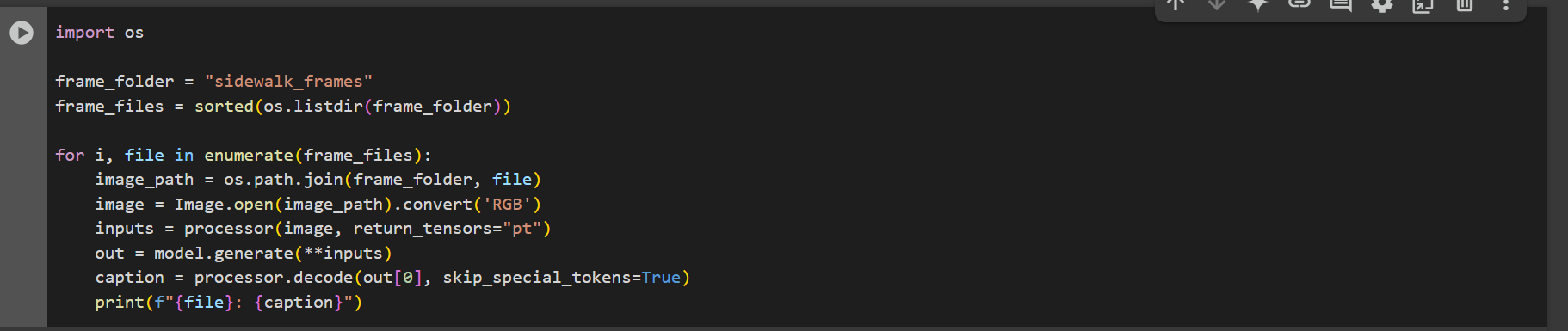
1. Open-CV Documentation - https://docs.opencv.org/
2. Hugging Face Transformers - https://huggingface.co/transformers/
3. PyTorch - https://pytorch.org/
4. BLIP Model - https://huggingface.co/Salesforce/blip-image-captioning-base
5. Google Colab - https://colab.research.google.com/

**5.10 Appendices**

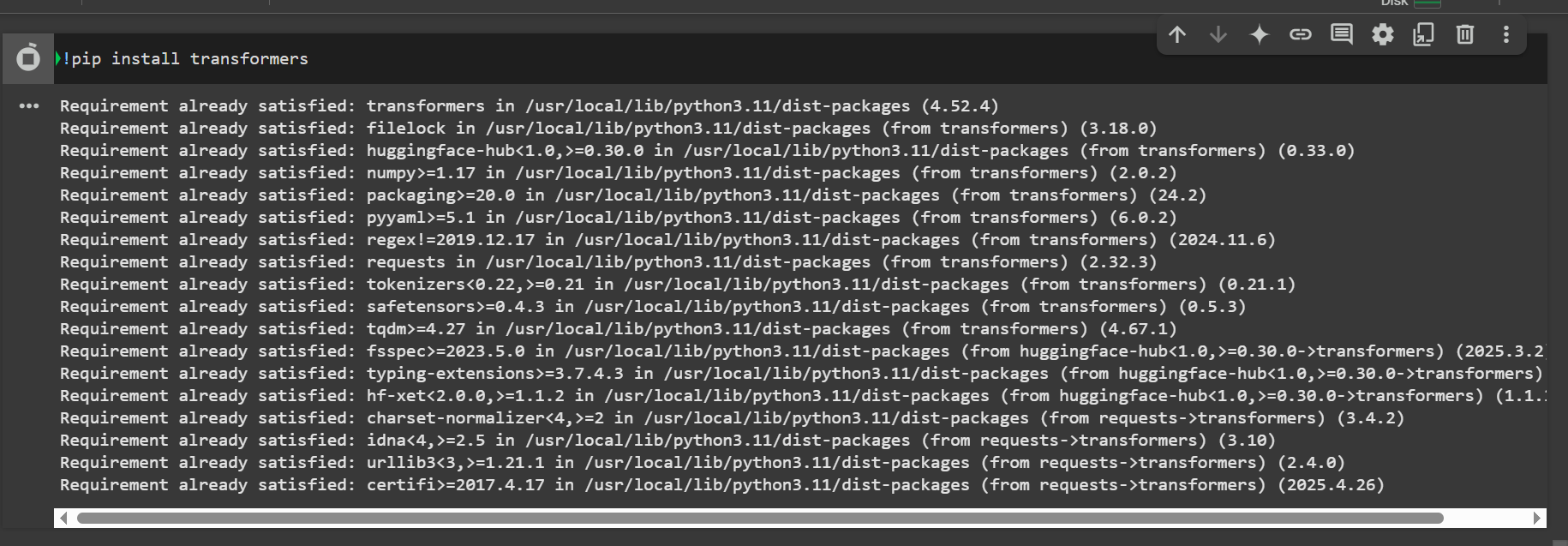
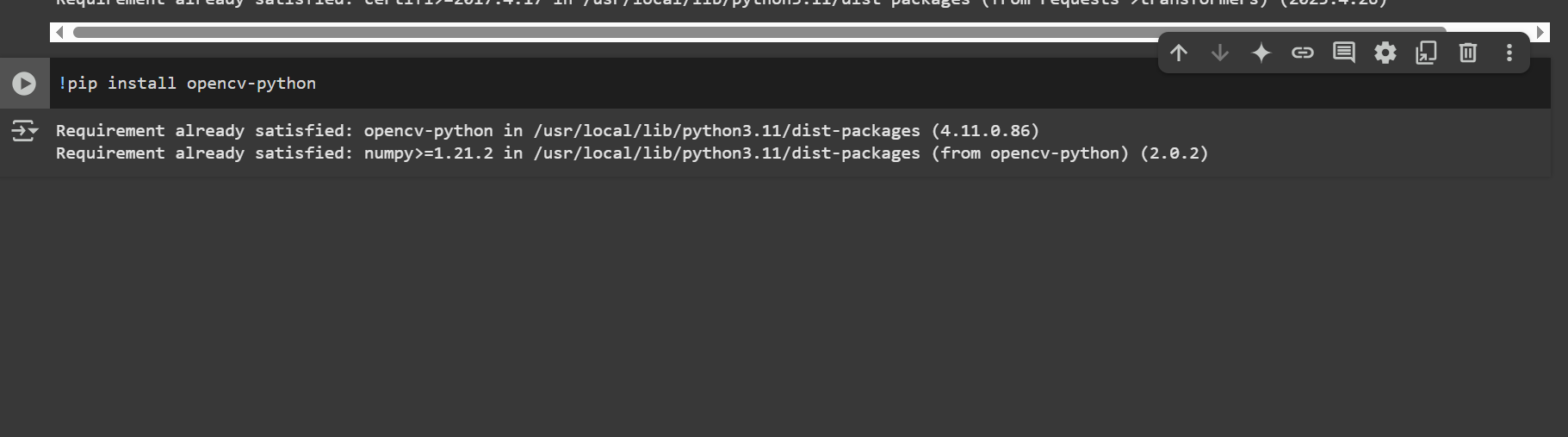
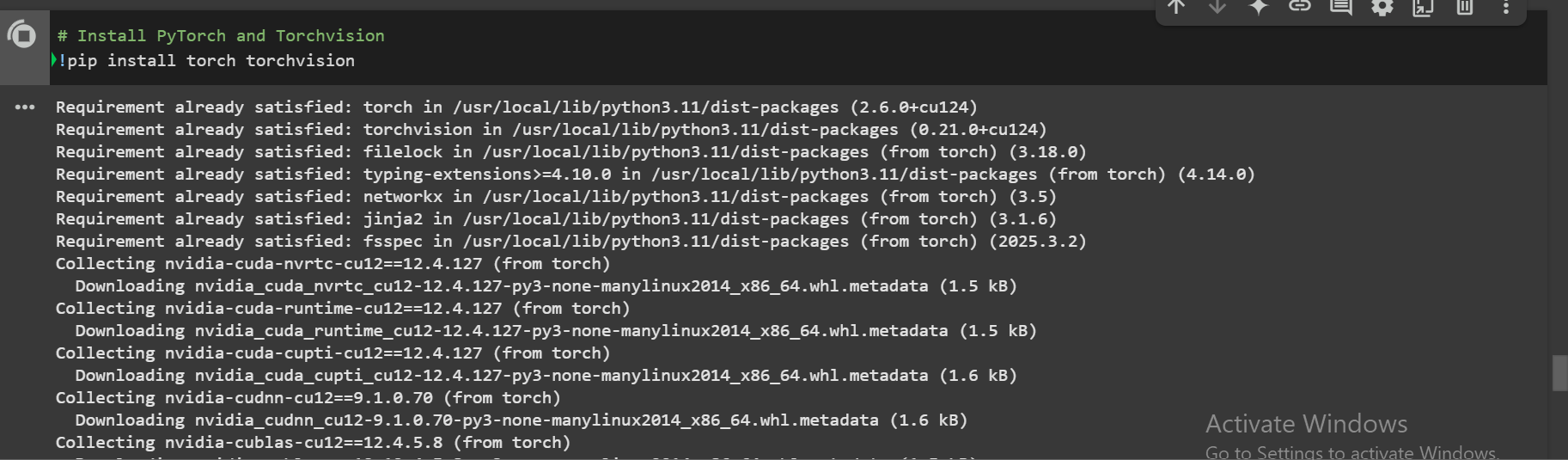
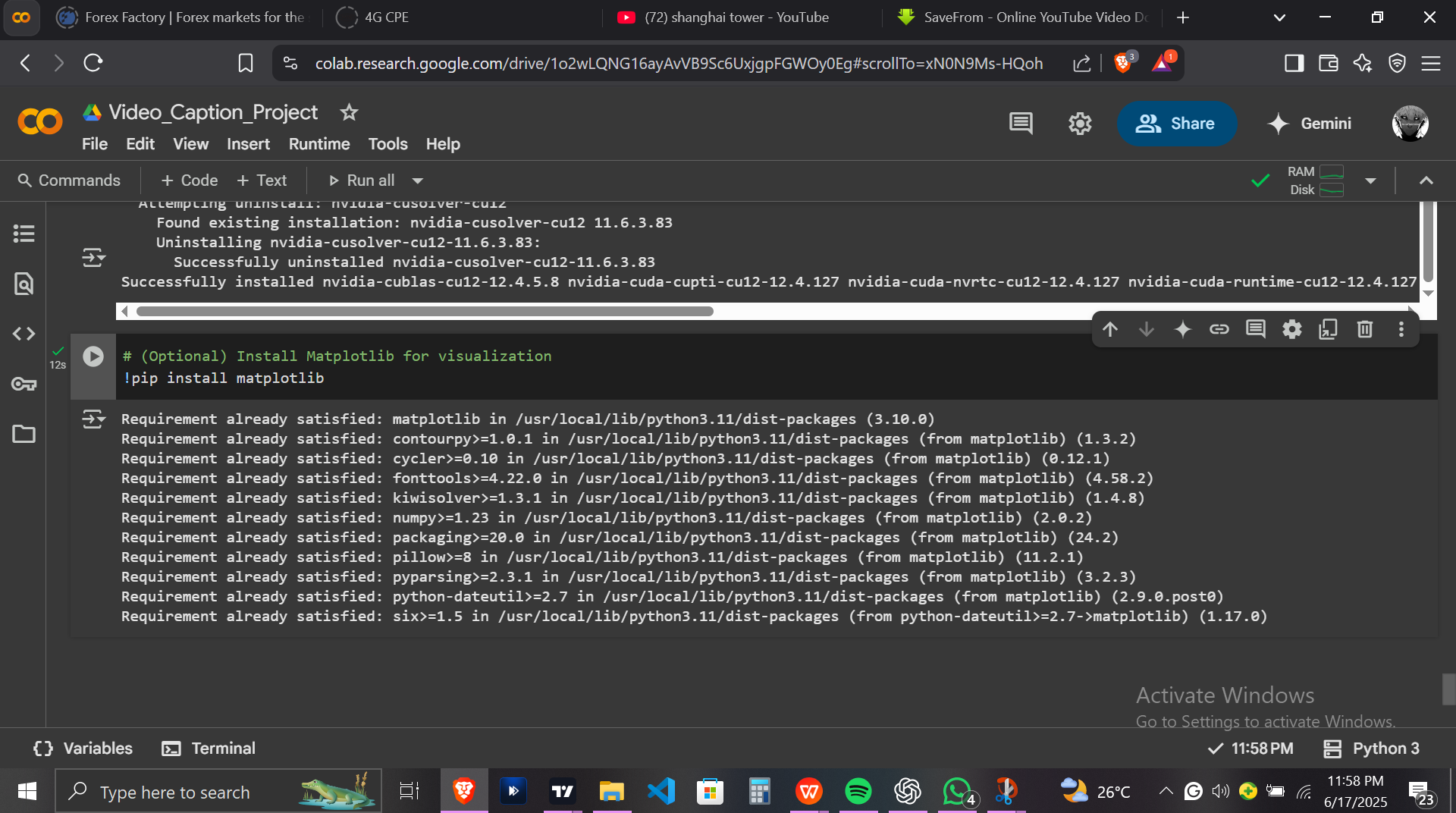
**Appendix A: Sample Output Captions**

* frame0.jpg: people walking down the street in london.
* frame1.jpg: A man in a white shirt

**Appendix B: Code Snippets**

* Frame Extraction Code
* Captioning Code
* Visualization Code

**Appendix C: Colab Environment Setup Commands**

* !pip install transformers
* !pip install opencv-python
* !pip install torch torch-vision
* !pip install matplotlib