

# **AI- ENHANCED VIDEO CAPTIONING USING VGG-16 AND CNN**

**A PROJECT REPORT**

*Submitted by*

**AYUSH PATTANAYAK [RA2111033010046]  
VASU RASTOGI [RA2111026010295]**

*Under the Guidance of*

**Dr. S. SADAGOPAN**

Associate Professor, Department of Computational Intelligence

*in partial fulfillment of the requirements for the degree of*

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE ENGINEERING**

**with specialization in Artificial Intelligence and Machine Learning**



**DEPARTMENT OF COMPUTATIONAL INTELLIGENCE  
COLLEGE OF ENGINEERING AND TECHNOLOGY  
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY  
KATTANKULATHUR- 603 203**

**MAY 2025**



Department of Computational Intelligence  
**SRM Institute of Science & Technology**  
**Own Work Declaration Form**

This sheet must be filled in (each box ticked to show that the condition has been met). It must be signed and dated along with your student registration number and included with all assignments you submit – work will not be marked unless this is done.

To be completed by the student for all assessments

**Degree/ Course** : B.Tech in Computer Science Engineering specialization in Artificial Intelligence and Machine Learning

**Student Name** : Vasu Rastogi & Ayush Pattanayak

**Registration Number** : RA2111026010295 & RA2111026010046

**Title of Work** : AI- Enhanced Video Captioning Using VGG-16 and CNN

We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.

We confirm that all the work contained in this assessment is our own except where indicated, and that We have met the following conditions:

- Clearly referenced / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

**DECLARATION:**

I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.

RA2111026010295  
RA2111026010046

AYUSH PATTANAYAK (RA2111033010046) VASU RASTOGI (RA2111026010295)



# SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR – 603 203

## BONAFIDE CERTIFICATE

Certified that 18CSP109L - Major Project report titled “**AI-Enhanced Video Captioning Using VGG16 AND CNN**” is the bonafide work of “**VASU RASTOGI [RA2111026010295], AYUSH PATTANAYAK [RA2111026010046]**” who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**SIGNATURE**

**Dr. S. SADAGOPAN**

SUPERVISOR

DEPARTMENT OF  
COMPUTATIONAL INTELLIGENCE  
SRM INSTITUTE OF SCIENCE AND  
TECHNOLOGY

**SIGNATURE**

**Dr. R. ANNIE UTHRA**

PROFESSOR & HEAD

DEPARTMENT OF  
COMPUTATIONAL INTELLIGENCE  
SRM INSTITUTE OF SCIENCE AND  
TECHNOLOGY

**EXAMINER I**

**EXAMINER II**

## ACKNOWLEDGEMENTS

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to **Dr. Leenus Jesu Martin M**, Dean-CET, SRM Institute of Science and Technology, for his invaluable support.

We wish to thank **Dr. Revathi Venkataraman**, Professor and Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We extend our sincere thanks to **Dr. M. Pushpalatha**, Professor and Associate Chairperson, School of Computing and **Dr. C.Lakshmi**, Professor and Associate Chairperson, School of Computing, SRM Institute of Science and Technology, for their invaluable support.

We are incredibly grateful to our Head of the Department, **Dr. R. Annie Uthra**, Professor, Department of Computational Intelligence, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our Project Coordinators, **Dr. M. Uma**, **Dr. M. S. Abirami**, Panel Head, **Dr. S. Sadagopan** and Panel Members, **Dr. R. Siva**, **Dr. P. G. Om Prakash**, **Dr. M. Salomi**, **Mrs. V. Kavitha**, **Mrs. Indhumathi** Department of Computational Intelligence, SRM Institute of Science and Technology, for their inputs during the project reviews and support.

We register our immeasurable thanks to our Faculty Advisor, **Dr. Maheshwari M**, Department of Computational Intelligence, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our guide, **Dr. S. Sadagopan** Department of Computational Intelligence, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under his mentorship. He provided us with the freedom and support to explore the research topics of our interest. His passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank all the staff and students of Computational Intelligence, School of Computing, S.R.M Institute of Science and Technology, for their help during our project. Finally, we would like to thank our parents, family members, and friends for their unconditional love, constant support and encouragement

Vasu Rastogi [RA2111026010295]  
Ayush Pattanayak [RA2111026010046]



# ABSTRACT

Video captioning is a rapidly evolving field in computer vision and natural language processing that involves generating textual descriptions of video content. With the explosive growth of multimedia data across platforms, there is a pressing need for systems that can interpret and describe visual information accurately and automatically. This research focuses on video captioning using Convolutional Neural Networks (CNN), particularly the VGG-16 architecture, to extract meaningful visual features that are used to generate coherent and contextually relevant captions. In the literature survey, we examine various approaches developed for video captioning, including template-based methods, statistical machine translation models, and deep learning frameworks involving RNNs and LSTMs. While early methods were constrained by limited vocabulary and rigid structures, modern neural models have improved performance but still face challenges in semantic understanding and temporal modelling. Many state-of-the-art systems integrate CNNs for visual feature extraction and recurrent architectures for sequence generation. Existing methodologies often utilize complex and computationally expensive models or require extensive labelled datasets. Some models lack the ability to generalize across diverse video content or struggle to maintain contextual consistency over longer sequences. Furthermore, the integration between visual and language modalities is often loose, limiting the depth of semantic interpretation. To overcome these limitations, we propose a video captioning framework that employs the VGG-16 CNN architecture for high-level feature extraction from key video frames, followed by a lightweight yet effective sequence modelling approach for caption generation. Keyframes are extracted at fixed intervals and passed through the pretrained VGG model to obtain deep feature representations. These features are then aggregated and fed into a decoder module—either an LSTM or Transformer-based network—which generates grammatically and contextually appropriate sentences. Experimental results were evaluated using benchmark datasets such as MSVD and MSR-VTT, with performance measured using BLEU, METEOR, and Cider scores. The proposed system demonstrated competitive results while maintaining lower computational complexity compared to heavier end-to-end models. The discussion includes an analysis of feature relevance, caption accuracy, and potential improvements using attention mechanisms or multimodal transformers. In conclusion, our VGG-CNN-based video captioning model provides a scalable, efficient, and interpretable solution to the challenge of automatic video understanding, making it suitable for real-world applications such as content indexing, assistive technology, and automated media tagging.

# TABLE OF CONTENTS

<b>ABSTRACT</b>	<b>v</b>
<b>TABLE OF CONTENTS</b>	<b>vi</b>
<b>LIST OF TABLES</b>	<b>viii</b>
<b>LIST OF FIGURES</b>	<b>ix</b>
<b>ABBREVIATION</b>	<b>x</b>

<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>10</b>
	1.1 GENERAL OVERVIEW	10
	1.2 PROBLEM STATEMENT	11
	1.3 MOTIVATION	11
	1.4 OBJECTIVES	12
	1.5 SCOPE	12
	1.6 SUSTAINABLE DEVELOPMENT GOAL	13
<b>2</b>	<b>LITERATURE SURVEY</b>	<b>14</b>
	2.1 RESEARCH GAPS	14
	2.2 RESEARCH OBJECTIVES	16
	2.3 EXISTING METHODOLOGY USED	16
<b>3</b>	<b>PROPOSED METHODOLOGY</b>	<b>18</b>
	3.1 FRAME EXTRACTION (SPRINT 1)	18
	3.2 FEATURE EXTRACTION USING VGG16 (SPRINT 2)	19
	3.3 LSTM CAPTION GENERATOR (SPRINT 3)	21

3.4 LSTM TRAINING MODEL	22
3.4.1 ENCODER MODEL	23
3.4.2 DECODER MODEL	24
<b>4 RESULTS AND DISCUSSION</b>	<b>25</b>
4.1 MODEL OUTPUT	25
4.2 MODEL EVALUATION	26
4.2.1 GREEDY SEARCH	26
4.2.2 BEAMSEARCH	26
4.3 EPOCH	28
4.3.1 LOSS	28
4.3.2 METRIC	30
<b>5 CONCLUSION AND FUTURE ENCHANCEMENTS</b>	<b>32</b>
5.1 CONCLUSION	32
5.2 FUTURE ENCHANCEMENTS	32
<b>5 REFERENCES</b>	<b>34</b>
APPENDIX	
A CODING	35
B PLAGIARISM	42
C CONFERENCE REPORT	46



## LIST OF TABLES

TABLE NO.	TITLE	PAGE NO.
2.1	Literature Survey	14
4.1	Greedy VS Beam Text	26
4.2	Epoch vs Loss	28
4.3.	Epoch vs Metric	30

## LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
3.1	Frame Extraction	17
3.2	Feature Extraction	18
3.3	LSTM Architecture	19
3.4	LSTM Training Model	21
4.1	Model Output	24
4.2	Graph for Epoch & Loss	30

## **ABBREVIATIONS**

<b>AI</b>	<b>Artificial Intelligence</b>
<b>LSTM</b>	<b>Long short term memory</b>
<b>SC</b>	<b>Serverless Computing</b>

# CHAPTER 1

## INTRODUCTION

### 1.1 General Overview

In today's digital era, video content has emerged as a dominant medium of communication, education, and entertainment. From online learning platforms and social media to corporate training and news dissemination, videos are widely consumed across the globe. However, a significant portion of this content remains inaccessible to individuals who are deaf or hard of hearing, non-native language speakers, and those in environments where audio playback is not feasible. This highlights the critical need for video captioning—an evolving technology that generates descriptive text reflecting both the audio and visual components of a video, including spoken dialogue, sound effects, and key visual cues.

Video captioning plays a pivotal role in ensuring digital inclusivity. By translating multimedia content into readable text, it not only enhances accessibility but also improves comprehension, engagement, and information retention among diverse audiences. The emergence of advanced technologies such as computer vision, machine learning (ML), and natural language processing (NLP) has significantly accelerated the development of intelligent captioning systems. These systems are now capable of analyzing video frames, recognizing speech and objects, and producing contextually appropriate captions in real time or during post-production.

The applications of video captioning extend across multiple industries including education, where it aids in inclusive learning; entertainment, where it supports diverse global audiences; and media archiving, where it enables efficient content indexing and retrieval. As digital content continues to proliferate, captioning technologies are becoming indispensable tools for bridging communication gaps and fostering a more inclusive online ecosystem. This paper delves into the technological foundations of video captioning, its current uses and significance, and the promising future driven by advancements in artificial intelligence. It further emphasizes the need to continue developing these systems to handle nuanced scenarios, improve contextual

understanding, and contribute meaningfully to accessibility and equality in the digital space.

## **1.2 Problem Statement**

Despite the rapid growth and consumption of video content across digital platforms, a substantial portion remains inaccessible to individuals with hearing impairments, non- native language speakers, and users in audio-restricted environments. The absence of accurate, real-time, and contextually relevant captions limits the inclusivity and effectiveness of video-based communication. Existing video captioning systems often struggle with challenges such as background noise, overlapping speech, complex visuals, diverse accents, and domain-specific terminologies. Moreover, many current solutions rely heavily on manual transcription, which is time-consuming, resource- intensive, and prone to human error. These limitations hinder the ability to scale captioning efforts effectively across diverse content types and languages. There is a growing need for automated, intelligent systems that can seamlessly integrate computer vision, machine learning, and natural language processing to generate high-quality captions in real time. Addressing these challenges is essential to ensure equitable access to information and foster a truly inclusive digital environment.

## **1.3 Motivation**

In the age of digital transformation, video content has become a dominant medium for communication, learning, and entertainment. However, this surge in video usage highlights a critical gap—accessibility for individuals who are deaf or hard of hearing, non-native language speakers, and people in environments where audio cannot be used. The lack of accurate and timely captions restricts access to information, marginalizes affected users, and undermines the inclusive potential of modern technology. The motivation behind advancing video captioning technology lies in the need to bridge this accessibility divide. Captions not only provide a textual representation of speech but also convey sound effects and visual context, making video content understandable and engaging for a broader audience. Additionally, captioning enhances comprehension, supports language learners, improves content indexing for search engines, and benefits viewers in noisy or sound-sensitive environments. With the evolution of artificial intelligence, machine learning, and natural language processing, there is immense

potential to create automated captioning systems that are fast, context-aware, and highly accurate. Investing in such technologies can significantly improve user experience, foster educational equity, and ensure that no one is left behind in the digital information age. This motivation underpins the drive to develop robust, scalable, and intelligent video captioning solutions.

## **1.4 Objective**

The primary objective of this study is to explore and analyze the technological advancements and methodologies involved in automated video captioning. This includes understanding how computer vision, machine learning (ML), and natural language processing (NLP) work together to identify speech, recognize visual elements, and generate accurate, contextually relevant captions in real-time or post-production. The study aims to assess the effectiveness of current captioning systems, identify existing challenges such as handling background noise, multiple speakers, and complex scenes, and evaluate the potential for multilingual and emotion-aware captioning. Additionally, the objective is to highlight the impact of video captioning in enhancing accessibility across various sectors—such as education, entertainment, media, and communication—while emphasizing the importance of inclusivity and digital equity. Ultimately, this work seeks to contribute to the development of more efficient, intelligent, and scalable captioning solutions for the ever-expanding landscape of digital video content.

## **1.5 Scope**

This study focuses on the design, development, and evaluation of automated video captioning systems powered by artificial intelligence, particularly computer vision, machine learning, and natural language processing. It covers both real-time and post-production captioning techniques and explores their application across various domains such as education, entertainment, online communication, and content management. The scope includes analyzing current tools and models, their capabilities in recognizing speech, identifying visual cues, and generating accurate and contextually meaningful captions. It also considers the integration of multilingual and adaptive features to accommodate diverse audiences. Furthermore, the study addresses the limitations of current systems, including issues related to background noise, speaker overlap, and contextual ambiguity. Ethical considerations such as fairness, accessibility, and cultural sensitivity are also within the scope. By outlining these aspects, the study aims to provide a comprehensive understanding of video captioning's role in creating an inclusive and accessible digital media environment.

## **1.6 Sustainable Development Goal**

The development and implementation of video captioning technologies strongly align with United Nations Sustainable Development Goal 10: Reduced Inequalities. This goal emphasizes the need to empower and promote the social, economic, and political inclusion of all individuals, irrespective of age, disability, language, or socio-economic background. Video captioning addresses a critical aspect of digital inclusion by making audio-visual content accessible to people with hearing impairments, language barriers, and those in environments where sound cannot be used. By ensuring that digital content is comprehensible and inclusive, video captioning contributes to reducing the information gap between different social groups. It also supports SDG 4: Quality Education, as captions enhance learning experiences for students, particularly those who rely on visual content or are learning new languages. In addition, captioning improves engagement and understanding in remote and hybrid learning environments, promoting inclusive and equitable quality education for all. As digital platforms continue to expand globally, integrating AI-powered captioning systems is a sustainable step toward an inclusive digital future. It ensures equal access to knowledge and communication, helping to build a world where everyone—regardless of ability or background—can fully participate in the digital ecosystem and benefit from technological advancements.

## CHAPTER 2

### LITERATURE SURVEY

#### 2.1 RESEARCH GAPS

SL.NO	TITLE	METHODOLOGY	RESEARCH GAP
1.	PWS-DVC: Enhancing Weakly Supervised Dense Video Captioning With Pretraining Approach	<p>Pretraining the decoder using caption supervision on the whole video before training it to describe individual events .</p> <p>Decodes event representations into textual descriptions using a transformer decoder.</p> <p>Learns event-level features by aggregating frame-level features using attention.</p>	<p>Existing methods lacked efficiency in generating accurate dense captions due to limited supervision.</p> <p>They ignored the utility of full video captions to guide dense video captioning.</p> <p>This paper addresses the gap by leveraging pretraining with whole-video captions, bridging the semantic mismatch between proposals and captions.</p>
2.	Video Index Point Detection and Extraction Framework Using Custom YOLOv4-Darknet Object Detection Model	<p>Proposed a custom YOLOv4-Darknet based system for detecting video index points (VIPs) for semantic video segmentation.</p>	<p>Lack of efficient methods for automated and precise index point detection in educational and instructional videos.</p> <p>Previous systems had issues with accuracy and real-time segmentation.</p> <p>This research addresses the need for an accurate, deep learning- based VIP detection method tailored to real-world videos.</p>
3.	SignBERT: A BERT-Based Deep Learning Framework for Continuous Sign Language Recognition	<p>Extracts frame-wise pose and hand features using OpenPose and 3D-CNNs.</p> <p>Encodes sequence of sign tokens with self-attention to capture context and temporal dependencies.</p> <p>Translates encoded features into glosses (text tokens representing sign units).</p>	<p>Traditional SL recognition methods relied heavily on frame- level classification and failed to model long-term temporal dependencies.</p> <p>There was no integration of strong NLP models like BERT to enhance semantic understanding.</p> <p>This study addresses the gap by applying BERT to the visual- linguistic domain of sign language, capturing both syntax and temporal order.</p>



4.	Q18-1013 (Early Action Prediction and Understanding Human Motion)	The authors use trajectory features and mid-level motion representations to predict human actions before they are fully executed.	The paper identifies a lack of effective methods for predicting actions before they are completed, rather than recognizing them after full execution.
		They focus on early action prediction, using partial observation of human motion sequences. Techniques such as Sparse Coding and Hidden Markov Models (HMMs) are used to model motion patterns and predict actions in progress.	Most prior research was on full-sequence action recognition; this work aims to improve proactive systems that must anticipate actions early (e.g., in surveillance or robotics).
5.	Video Action Understanding	This paper surveys a wide range of video action recognition techniques, particularly deep learning-based models.  It categorizes existing approaches into two-stream networks, 3D CNNs, spatiotemporal models, and attention mechanisms.	The paper highlights scalability and generalization as major challenges. A key gap is the limited ability of current models to generalize across domains, tasks, and datasets.  It also notes the lack of explainability in deep models and calls for better understanding of how and why models make certain predictions.
6.	Human Action Recognition: A Survey (Video Action Understanding by Aggarwal et al.)	This survey provides a historical and structural overview of handcrafted features (e.g., STIPs, HOG3D) vs. deep learning models for human action recognition.	The authors emphasize challenges like viewpoint variation, occlusion, intra-class variability, and real-time inference.
		It classifies methods based on whether they are single-view or multi-view, 2D or 3D, and whether they use pose-based, appearance-based, or motion-based features.	They point out the difficulty of action recognition in unconstrained, real-world environments as an underexplored issue.
			The paper also highlights the need for real-time, lightweight models deployable in edge devices.
7.	"Development of an End-to-End Deep Learning Framework for Sign Language Recognition, Translation, and Video Generation	Uses Convolutional Neural Networks (CNNs) such as ResNet for spatial feature extraction from video frames.  Follows with Bidirectional Long Short-Term Memory (Bi-LSTM) networks for temporal sequence modeling, allowing the model to capture motion dynamics in sign gestures.  GANs are trained to produce visually realistic videos that simulate a person performing the corresponding signs.	Most datasets used are constrained to specific sign languages (like American Sign Language), limiting the model's generalizability to other regional or national sign languages.  While the framework achieves good accuracy, its computational complexity

			makes real-time performance difficult, especially in video generation.
--	--	--	--

## 2.2 RESEARCH OBJECTIVES

Across these seven studies, the overarching goal is to bridge visual and linguistic modalities in video-based communication and understanding, each tackling a distinct yet complementary challenge. Sign BERT aims to advance continuous sign language recognition by integrating a (3+2+1)D ResNet for spatial feature extraction with a BERT encoder—pretrained via masked isolated-sign data—to capture temporal and linguistic dependencies, further enhancing robustness through a multimodal hand-image alignment mechanism and a BLSTM decoder. The H-DNA framework extends sign-to-text work by providing a fully end-to-end pipeline: Media Pipe-driven pose extraction and a hybrid CNN+Bi-LSTM model recognize signer gestures in real time, an NMT module translates these into natural-language sentences, and a Dynamic GAN generates photorealistic sign-language video from text, thereby enabling bidirectional communication. In the domain of video captioning, the Multi-Faceted Attention model proposes a unified LSTM-based architecture with parallel attention branches over temporal frames, motion cues, and semantic tags; by learning to focus on the most relevant features for each output word, it narrows the vision-language gap and achieves state-of-the-art results on MSVD and MSR-VTT. Finally, PWS-DVC addresses the scarcity of densely annotated event-caption corpora by leveraging abundant clip-level caption datasets to pretrain an event-captioning module, then fine-tuning it together with a weakly supervised caption-localization module on Activity Net Captions; this strategy obviates the need for explicit event boundaries, improves both localization and caption quality, and reduces annotation overhead. Collectively, these works push toward more accurate, robust, and scalable video-to-language and language-to-video systems, with applications ranging from accessibility for the deaf community to automated video indexing and retrieval.

## 2.3 EXISTING METHODOLOGY USED

The existing methodology for AI-enhanced video captioning typically involves a two-stage process combining feature extraction from video frames and natural language generation to produce descriptive captions. At the core of this system lies the use of Convolutional Neural Networks (CNNs), particularly VGG16, a pre-trained deep CNN model that excels in visual feature extraction. VGG16 is employed to extract spatial features from key video frames. Videos are first sampled at regular intervals to select representative frames, which are then resized and fed into the VGG16 model. The output from one of the final fully connected layers (often FC2) represents high-level visual features that encapsulate the content of each frame. These extracted features are then passed into a sequence modelling network, often implemented using Recurrent Neural Networks (RNNs) or their advanced versions like Long Short-Term Memory (LSTM) networks or Gated Recurrent Units

(GRUs). These models are effective in learning temporal dependencies between frames, enabling the generation of coherent sentences. In recent approaches, the CNN acts as the encoder to transform image data into fixed-length feature vectors, while the RNN acts as the decoder that translates these features into human-like captions. Attention mechanisms may also be incorporated to dynamically focus on relevant parts of the video during different stages of sentence generation, improving caption accuracy. For training such models, datasets like MSVD (Microsoft Video Description Corpus) or MSR-VTT are used, where each video is annotated with multiple human-generated captions. The training process involves minimizing the difference between the predicted captions and ground truth using a loss function such as categorical cross-entropy. During inference, techniques like beam search may be employed to generate the most likely caption sequence. In terms of software, popular frameworks such as TensorFlow or PyTorch are used to implement the deep learning architecture. Preprocessing steps include frame extraction, resizing, normalization, and tokenization of text data. The integration of VGG16 ensures that the model benefits from transfer learning, improving performance even with limited labelled data. This methodology offers a robust pipeline for generating meaningful and context-aware video captions, playing a vital role in applications like video indexing, accessibility tools, and intelligent content retrieval systems.

## CHAPTER 3

### PROPOSED METHODOLOGY

#### 3.1 FRAME EXTRACTION (SPRINT 1)

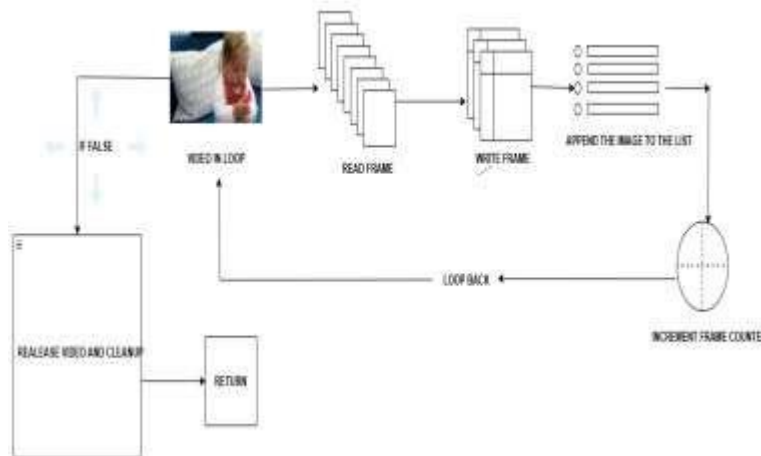


Figure 3.1 Frame Extraction Diagram

Feature Extraction Diagram illustrates a video processing workflow that involves reading and processing video frames sequentially. The process begins by opening a video file and checking whether it is accessible. In fig3.1 if the video file is unavailable or unreadable, the system releases the video resource and performs cleanup before terminating the operation. If the video file is accessible, the system enters a loop where it continuously reads frames from the video. Each frame is processed and written, after which the processed frame is appended to a list for further use, such as analysis or creating a modified video file. A frame counter is incremented after each processed frame to track progress. The loop continues until all frames are processed. Once the video ends, the system releases the video resource and performs cleanup operations to ensure efficient memory management before terminating the process. This workflow is typical in video processing tasks like object detection, frame-by-frame analysis, and video enhancement.

### 3.2 FEATURE EXTRACTION USING VGG16 (SPRINT 2)

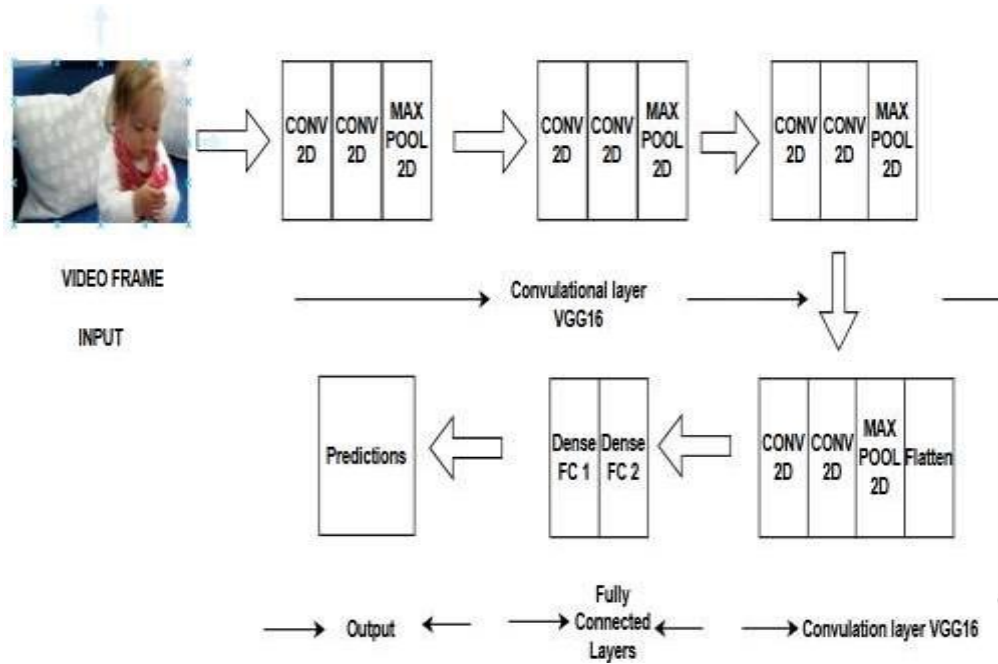


Figure 3.2 Feature Extraction Diagram

The Feature Extraction Diagram illustrates a pipeline of deep learning dedicated to feature extraction from the VGG16 convolutional neural network (CNN), used particularly for examining frames pulled from a video. The operation starts with a video frame input, which is used as raw image data for the model. The image is applied to multiple convolutional and pooling layers, which constitute the basis of the feature extraction process. During the feature extraction process, every Conv2D (convolutional) layer filters the image to identify local patterns like edges, textures, and shapes. The filters move across the image, firing when they recognize certain patterns, essentially pointing out important features in various areas of the frame. MaxPooling2D layers, following the convolution layers, shrink the spatial size of the feature maps maintaining the most important information, which reduces computation and overfitting. In fig3.2 the combination of convolution and pooling processes is repeated in various stages so that the network extracts more and more abstract and higher-order features as it proceeds deeper into the architecture. These layers together form the VGG16 model's convolutional base, whose sole purpose is to extract the features from the input image. After extracting high-level features, the data flows through a Flatten layer, where the multi-dimensional feature maps get flattened into one-dimensional vector space. This output, being a flattened representation of the rich features learned from the image, then gets fed to the fully connected layers for interpretation. The fully connected (dense) layers perform the classification. They receive the

extracted features and learn to project them to the appropriate output classes—this is where the decision is made. The final prediction layer then outputs the result based on the extracted features in the previous stages.

### 3.3 LSTM ARCHITECTURE CAPTION (SPRINT 3)

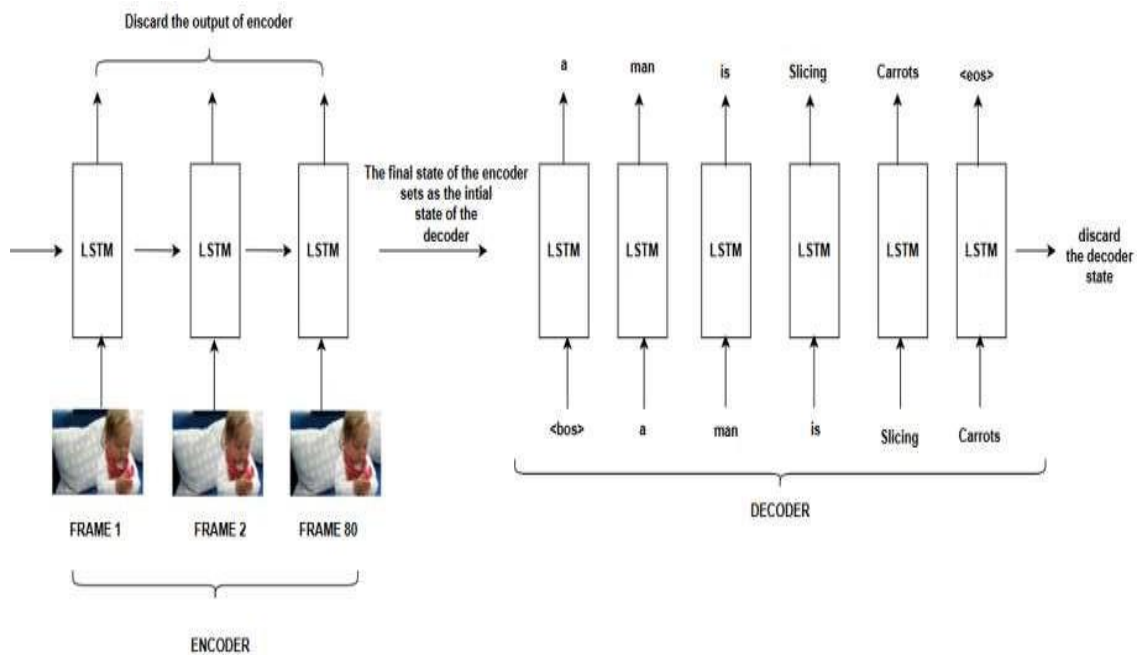


Figure 3.3 LSTM Architecture

LSTM Architecture illustrates a video caption generation model designed to describe new videos after being trained. It follows the Encoder–Decoder framework using LSTM (Long Short-Term Memory) networks. The process starts with a video composed of multiple frames, such as Frame 1 to Frame 80. Each of these frames has already gone through feature extraction using a system like VGG16, resulting in a feature vector for each frame. These feature vectors, representing each frame, are processed one by one by the encoder. The encoder consists of LSTM units that understand the sequence and how frames relate to each other over time. Unlike other models that use outputs from every step, this model only keeps the final hidden state from the encoder. This final hidden state acts as a summary of all the information from the video segment. This summary, known as the context vector, initializes the decoder, which is another set of LSTM cells. The decoder's job is to create the caption. It starts with a special token called <bos> (beginning of sentence), indicating the start of the caption. At each step, the decoder uses the previous word to predict the next one. For instance, for a

video depicting a man slicing vegetables, the decoder might produce the sentence: “a man is slicing carrots.” It begins with <bos>, then predicts “a”, then “man”, and continues in this manner. Each predicted word feeds into the subsequent LSTM unit in the decoder, creating a sequence of words. This process continues until the model produces the <eos> (end of sentence) token, which signifies that the caption is complete. Once the caption is finished, the internal state of the decoder is discarded. In fig 3.3 this model effectively converts video sequences into meaningful sentences by utilizing the strengths of LSTMs to manage sequential data. It captures image features through CNN-based extraction and time-related dynamics using LSTM encoders and applies these to language creation with the decoder.

### **3.4 LSTM TRAINING MODEL**

This model explains how a special model is trained to make captions for videos using a system called an LSTM-based Encoder–Decoder with Beam Search. In fig3.4 this model learns by looking at pairs of videos and their captions to create accurate and meaningful explanations. Training starts with a sequence of video frames, like from Frame 1 to Frame 80. These frames have already been processed by a CNN to get important features. Each frame's features are sent one by one into the encoder. The encoder is made of LSTM units, which help the model understand the order and timing in the video. Each LSTM takes the features of one frame, processes it, and passes the information to the next LSTM. This builds a detailed understanding of the whole video.



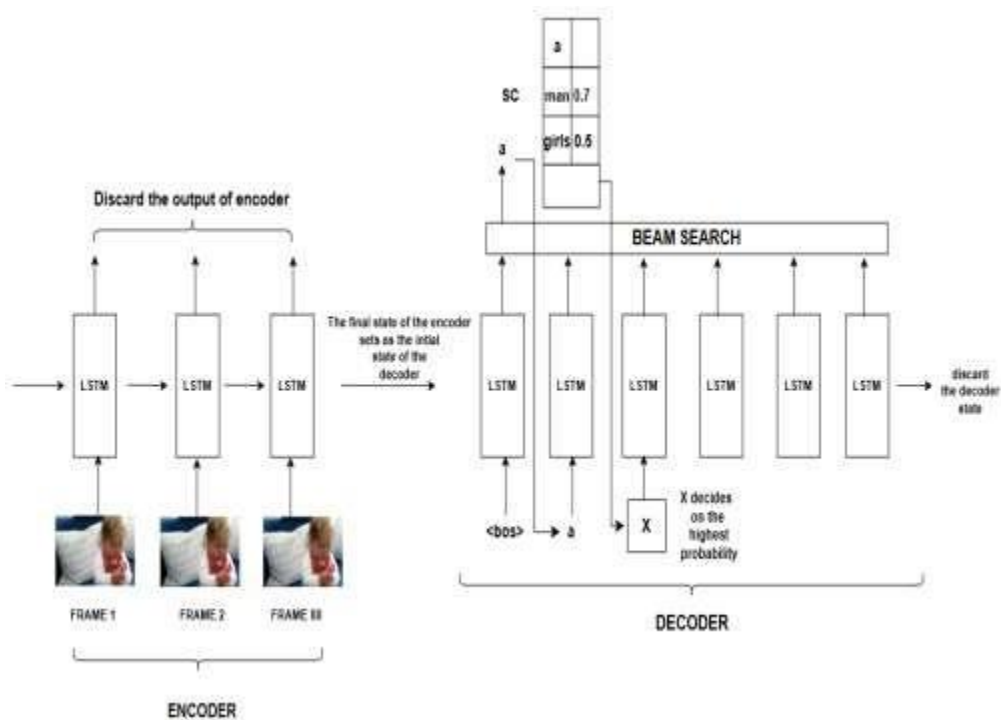


Figure 3.4 LSTM Training Model

During this, the middle outputs from the encoder LSTMs are ignored. Only the final information, which includes the video context, moves to the decoder. The decoder, also made of LSTM units, starts making the captions. It begins with a special token called <bos> (beginning of sentence) and is trained to guess the next words using the earlier words and the video context. The model uses a method called Beam Search instead of picking just the most probable word each time. Beam Search keeps several word options open at every step instead of deciding on just one right away. For instance, after starting with “a,” it might consider “man” with a probability of 0.7 and “girls” with 0.5. Both options remain for more exploration. A part of the model known as “X” decides which word has the highest probability, guiding the formation of a likely sentence. The decoder keeps picking words one by one until it gets to the token <eos> (end of sentence) or hits a set limit on sentence length. The model looks at the predicted caption and the real one to find mistakes. These mistakes help the model improve through a process called backpropagation, which adjusts the model. This whole setup helps the model turn video frames into meaningful text captions. The Beam Search method helps choose better-quality captions. This training process fine-tunes the model for accuracy, smooth writing, and context relevance, preparing it for real-life use.

### 3.4.1 ENCODER MODEL

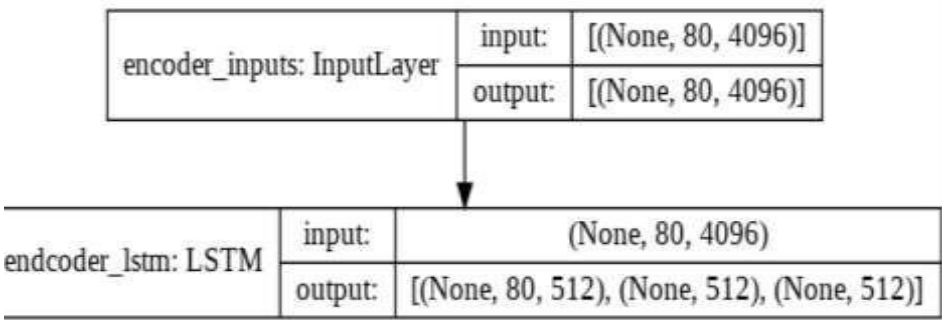


Figure 3.5 Encoder model

The picture explains how the encoder in a video captioning model works. It shows how video frames are processed one after the other using something called an LSTM (Long Short-Term Memory) layer. The data given to the encoder, known as encoder\_inputs, has a shape described as (None, 80, 4096). 'None' refers to the batch size, which can change, '80' indicates 80 frames from the video are used, and '4096' is the size of the feature vector for each frame. These feature vectors are usually created with networks like VGG16, which have been trained already. They help capture the details and characteristics of each frame in the video.

### 3.4.2 DECODER MODEL

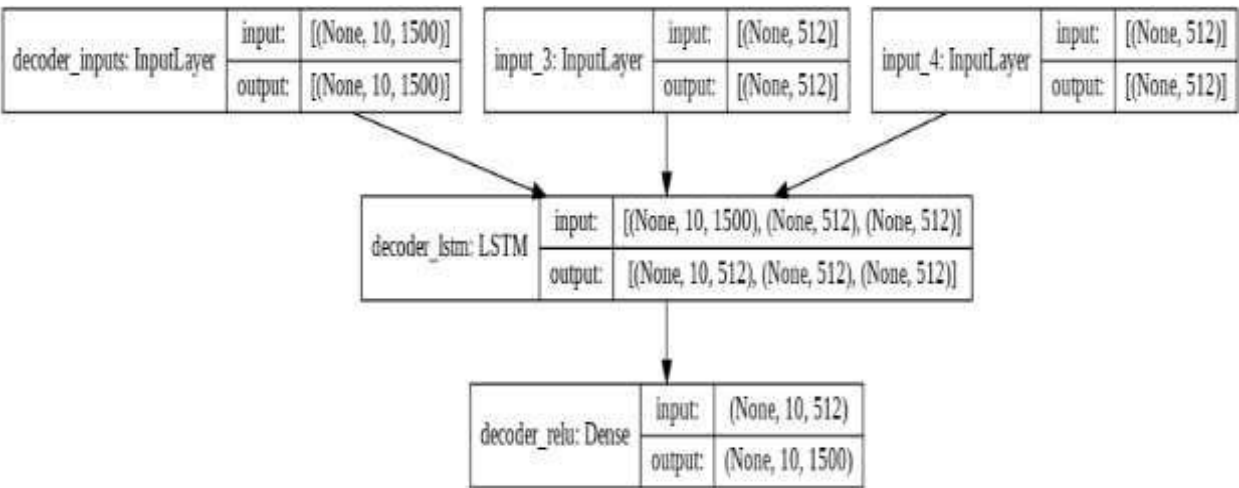


Figure 3.6 Decoder Model

In fig 3.6 it shows how the decoder part of a video captioning model operates. This model's job is to

create a series of words, forming a caption, from video features that have been processed by the encoder. The decoder is made up of two main parts: an LSTM layer and a Dense layer, which is also called a fully connected layer. These parts work together to start generating the caption. The primary input to the decoder is called `decoder_inputs`, which has a specific shape: `(None, 10, 1500)`. Here, "None" is a placeholder for the number of video clips being processed at once, 10 represents the number of steps in time, which is also the maximum length for a caption or how many words you want to generate, and 1500 refers to the complexity level or the size of each word's representation. These representations can be acquired from an embedding layer or pre-trained word vectors. There are three more inputs needed for the decoding process. They are named `input_3`, `input_4`, and one input without a name, each with a shape of `(None, 512)`. These inputs are significant because they represent the initial hidden state and cell state of the LSTM decoder. They play a crucial role in guiding the generation of the word sequence. Typically, these states are the final outputs from the encoder LSTM, and they are essential for maintaining the order and timing of words in the caption, ensuring the caption makes sense logically and temporally.

## CHAPTER 4

### RESULTS AND DISCUSSION

#### 4.1 MODEL OUTPUT

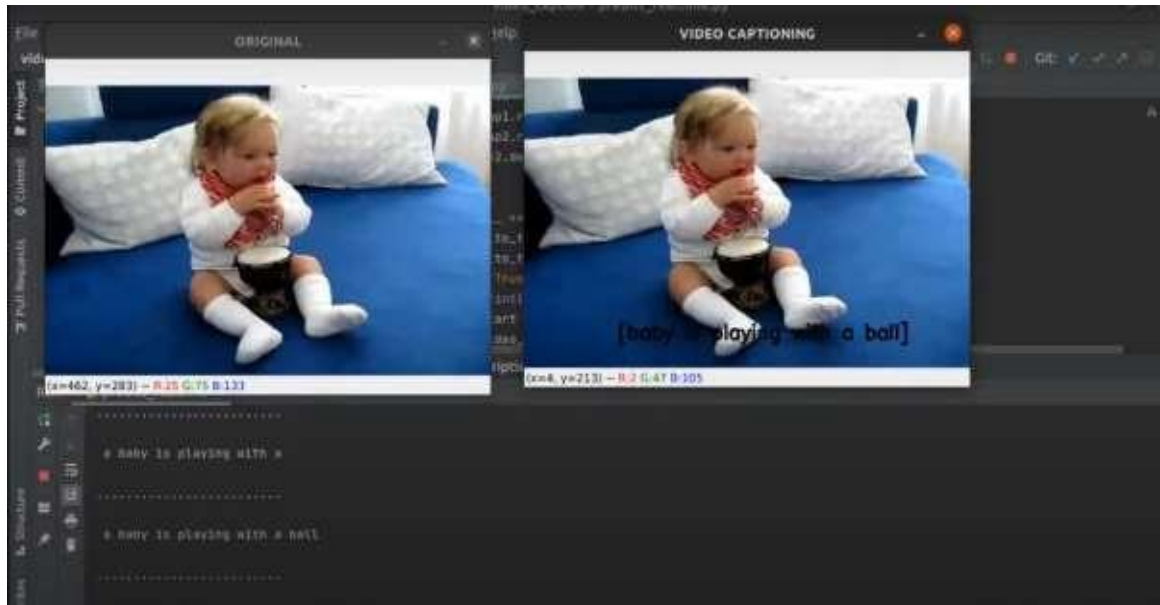


Figure 4.1 Model Output

The video captioning system shows how strong the deep learning model is, especially since it uses an LSTM-based encoder-decoder structure. This system takes videos, which is hard to do, and turns them into clear text descriptions. It uses Long Short-Term Memory (LSTM) networks, which are very good at dealing with data that comes in sequences and holding onto information over time. One big plus is that the model can grasp what's happening in a scene. For instance, it can see that a baby is playing with something. Even if it makes a small mistake like naming an object incorrectly, the caption it creates is still meaningful and correct, showing that it knows how to create smooth sentences in natural language. When it gets a full sentence correct, like saying “a baby is playing with a ball,” it proves that using a beam search algorithm works well in decoding. This method picks the most likely words based on word probabilities it has learned. In fig 4.1 the system works in real-time, which means it can show video and captions together, making it very useful for live video analysis, accessibility tools, and automated monitoring. In fig4.1 the system shows that using techniques like LSTM for sequence modeling, understanding context, and beam search for decoding can make logical and accurate captions for videos. This example illustrates the effective combination of computer vision and language processing using today's deep learning technology.

## 4.2 MODEL EVALUATION

### 4.2.1 Greedy Search

Greedy Search is a basic and quick method used in tasks like video captioning to generate sentences that describe the input. The process begins with a special token called the beginning-of-sequence token ( $\text{<S>}$ ). At each step, the model selects the word with the highest probability from the vocabulary, as determined by the decoder. This continues, word by word, until it reaches an end-of-sequence token ( $\text{<E>}$ ) or a set maximum length. The major advantage of Greedy Search is its simplicity and low need for computing power, making it ideal for quick results like in real-time applications. However, it only chooses the best word at each step and doesn't plan ahead, which can lead to missing better overall sentence options. This can result in captions that are not the best or are repetitive. Despite these limitations, Greedy Search is still useful in many situations. It's particularly helpful in the early stages of testing or when there are limited resources available.

### 4.2.2 Beam Search

Beam Search is a technique used to generate sentences in tasks such as video captioning, where it is very important for the sentences to be clear and fit the context well. This method is different from Greedy Search, which selects only the most likely word at each step. Beam Search, on the other hand, keeps several potential sentence paths open at once. These paths are known as "beams," and the number of beams is determined by something called beam width ( $k$ ). At every stage in building a sentence, Beam Search considers all the possible next words. It expands the sentence paths and retains only the top  $k$  paths based on their likelihood of being correct. This method allows the system to consider many different options instead of locking into a single choice too soon. The major strength of Beam Search is its ability to strike a good balance between exploring various possibilities and choosing the most optimal result. This often results in generating better, more natural sentences. For example, when used for video captioning, Beam Search can produce a complete, grammatically correct sentence that offers a more accurate description of what is shown in the video. It does this by weighing multiple word combinations carefully before picking the best one.

SAMPLE FRAMES	BEAM TEXT	GREEDY TEXT
	a man is mixing ingredients in a bowl  (38.16s)	a man is mixing ingredients in a bowl  (0.69s)
	a cat is playing the piano (26.48s)	a cat is playing the piano (0.70s)
	a person is cutting a pineapple (24.31s)	a person is cutting a piece of pieces  (0.70s)
	a woman is mixing some food (35.91s)	a woman is mixing some food (0.72s)

Table 4.1 Comparison for Greedy & Beam Text

In table 4.1 it carefully compares Beam Search and Greedy Search by analyzing the captions they produce for different video frames. It highlights the strengths and weaknesses of each method. These two strategies help us understand how they work for creating video captions in real-world situations. Greedy Search is quite simple. It chooses the word that seems most likely at each step of making a caption. This simplicity makes it fast and easy to use, but the downside is that it sometimes misses the overall meaning of the sentence. For example, Greedy Search might generate a sentence like: “a person is cutting a piece of pieces.” This example shows a lack of clarity and is repetitive. It highlights the main problem with Greedy Search: it often picks words that seem right individually but don't fit well together as a complete sentence. Beam Search, in contrast, uses a more advanced approach. It considers multiple possible word sequences at each step, looking at various combinations before deciding on the final words. This allows Beam Search to create captions that match the scene more accurately. For instance, in the same situation, Beam Search might correctly generate: “a person is cutting a pineapple.” This sentence is both clear and accurate, showcasing Beam Search's ability to handle complicated scenes where understanding details correctly is important. In scenes that are simpler, both Greedy and Beam Search can perform equally well. For example, if a video shows a woman mixing food, both methods might result in the sentence: “a woman is mixing some food.” This indicates that when the scene is straightforward, Greedy Search can still deliver acceptable results. Greedy Search offers the benefits of speed and ease of use but sometimes falls short in quality. Beam Search, while requiring more computational power, usually creates better sentences that sound natural and are precise. This makes Beam Search a preferred choice for applications like video captioning, where the richness and reliability of the text are crucial for clear and accurate interpretation.

## **4.3 EPOCH**

In deep learning for video captions, an "epoch" means a full cycle through the training data. During an epoch, the model processes all the training samples and updates its internal settings. This helps it learn to generate better captions for videos. Training involves many epochs, allowing the model to improve its predictions over time. After each epoch, the model fine-tunes itself using feedback from the previous pass to become more accurate and better at handling new data. If there are too many epochs, the model might become too focused on the training data and might not perform well on new data. This issue is called "overfitting." To avoid overfitting, techniques like observing validation results or stopping the training early are used to find the optimal number of epochs. So, epochs are crucial as they help the model gradually strengthen its ability to connect video content with natural language descriptions.

### 4.3.1 LOSS

Loss is an important concept in training a model. It measures how different the model's predicted captions are from the real ones written by humans. A common type of loss is cross-entropy loss. This evaluates the probabilities the model assigns to each word. If the model assigns low probabilities to the right words, it receives a bigger penalty. Loss for each word is calculated and then summed up or averaged for the whole caption. The goal of training is to lower this loss so the model improves at generating captions similar to those written by humans. However, humans do not evaluate captions word by word; they look at the overall meaning and context. To improve this, advanced training methods like reinforcement learning use evaluation tools such as CIDEr or METEOR in the loss function. These tools help make the model optimize better for human-like captions. Therefore, loss in video captioning is key because it guides the learning process and helps the model get better at creating fluent and meaningful descriptions.

<b>Epoch (X-axis)</b>	<b>Train Loss (Y-axis)</b>	<b>Test Loss (Y-axis)</b>
0	4.2	3.8
20	1.5	1.6
40	1.0	1.2
60	0.8	1.05
80	0.72	1.00

Table 4.2 Epoch VS Loss



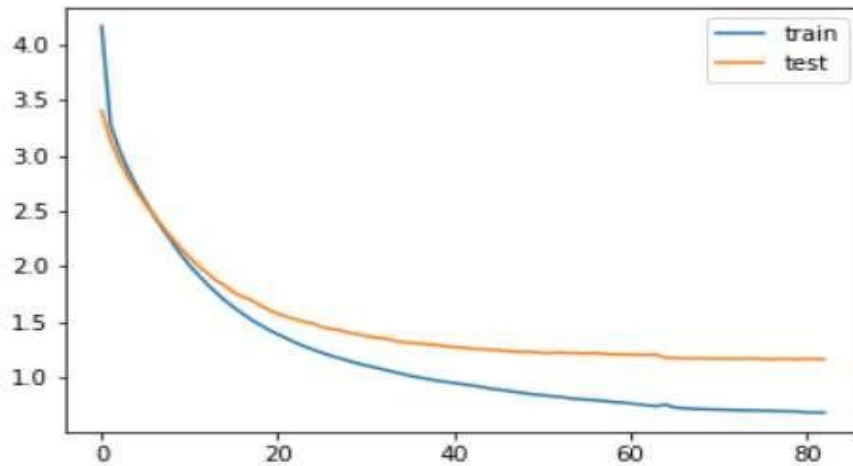


Figure 4.2 graph for Epoch & Loss

The table shows changes in training and test loss values as a video captioning model learns over time. Initially, at epoch 0, we see the training loss is 4.2 and the test loss is 3.8. These high numbers indicate the model hasn't learned much yet. As the model continues, by epoch 20, the training loss decreases to 1.5, with the test loss closely following at 1.6. This suggests the model is improving. By epoch 40, the training loss drops to 1.0, and the test loss is 1.2. Progressing to epoch 60, the training loss is 0.8, and the test loss is 1.05. Finally, by epoch 80, the training loss reduces further to 0.72, and the test loss levels out at 1.00. The consistent drop in both training and test losses shows effective learning. The small gap between them means the model isn't overfitting and can handle new, unseen data well. The model displays a reliable learning pattern with both training and test losses steadily going down with each epoch. The close values of training and test losses point to its strong performance in generalizing, making it dependable for generating captions on new video content.

### 4.3.2 METRICS

Metrics are tools used to see how good automatically created captions are by comparing them to captions written by people. These metrics check if the sentences are clear, correct, and make sense. Some well-known metrics are BLEU, which looks at parts of sentences; METEOR, which examines precision, recall, and synonyms; ROUGE, which focuses on how much overlap there is between texts; and CIDEr, which considers how often words appear in different captions. Each of these metrics highlights different parts of what makes a good caption, so using more than one gives a better overall picture of a model's performance. While these metrics give scores, they don't always match what a person might think, so it's important to also review the captions by actually looking at them.

Epoch (X-axis)	Train Acc (Y-axis)	Test Acc (Y-axis)
0	0.35	0.37
20	0.68	0.65
40	0.76	0.71
60	0.79	0.72
80	0.80	0.73

Table 4.3 Epoch vs Metrics

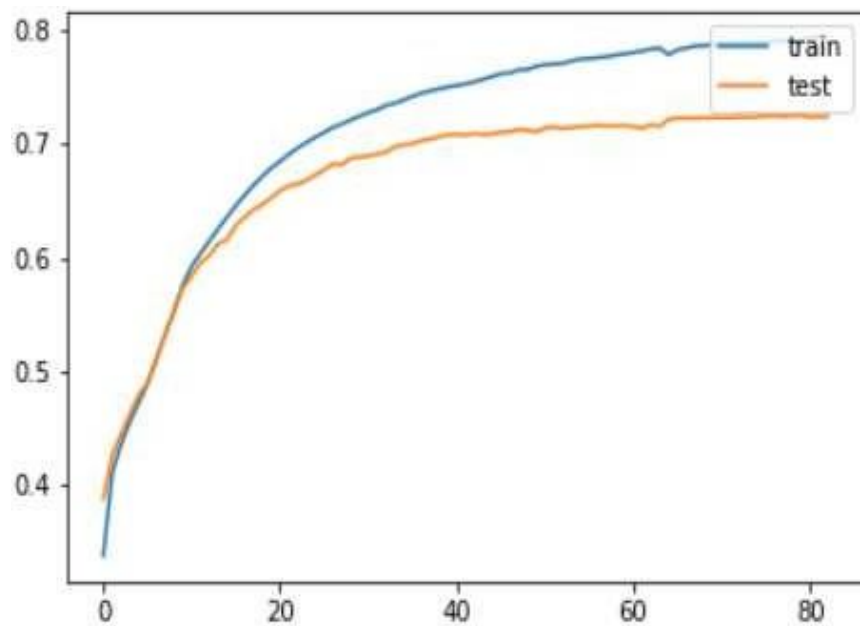


Figure 4.3 graph for Epoch & Metric

The table shows how a video captioning model's accuracy changes over different times, called epochs. On one axis, we have the number of epochs, and on the other, we have accuracy scores. When we start at epoch 0, the model isn't doing very well. The training accuracy is 0.35, and the test accuracy is 0.37, which means the model is in its early learning stages and makes many mistakes. By the time we reach epoch 20, we see improvements. The training accuracy climbs to 0.68, while the test accuracy rises to 0.65. This is a sign that the model is picking up important patterns. Fast forward to epoch 40, and the training accuracy improves to 0.76, with the test accuracy moving up to 0.71. The model continues to learn and get better. At epoch 60, the training accuracy reaches 0.79, and the test accuracy is 0.72. Finally, by epoch 80, the training accuracy hits 0.80, and the test accuracy is 0.73. After epoch 40, the test accuracy increases more slowly, suggesting that the model is getting close to its best performance levels. The model shows a strong improvement in both training and test accuracy, especially in the earlier epochs. By epoch 80, achieving a training accuracy of 80% and a

test accuracy of 73% means the model has learned effectively and can handle new data well. The small gap between training and test accuracy indicates that there's minimal overfitting, making this model trustworthy for real-world video captioning jobs.

## **CHAPTER 5**

### **CONCLUSION AND FUTURE ENHANCEMENT**

#### **5.1 CONCLUSION**

Video captioning has emerged as a crucial technology for enhancing digital accessibility and inclusivity. By converting complex audio-visual information into coherent text, it bridges communication gaps for people with hearing impairments, non-native speakers, and users in audio-restricted environments. The advancement of computer vision, machine learning, and natural language processing has enabled the development of intelligent captioning systems that accurately interpret speech, recognize visual cues, and generate contextually relevant captions.

Recent innovations, such as transformer-based architectures and attention mechanisms, have significantly improved the quality and fluency of generated captions by focusing on important spatial and temporal features. The introduction of large-scale multimodal datasets and self-supervised learning has further accelerated progress, minimizing reliance on manual annotations.

Looking ahead, video captioning holds immense potential in real-time multilingual translation, emotion recognition, and smart summarization, broadening its applications across assistive technologies, education, surveillance, and entertainment. However, ethical challenges like bias, privacy, and cultural sensitivity must be carefully addressed to build trustworthy and fair systems.

In conclusion, video captioning stands as a powerful intersection of AI disciplines, playing an essential role in shaping an inclusive, intelligent, and accessible digital future. Continued innovation will ensure that digital content becomes universally understandable and truly global.

#### **5.2 FUTURE ENCHANCEMENT**

The future of video captioning is poised for significant advancements driven by rapid innovations in artificial intelligence, machine learning, and natural language processing. One key area of enhancement is the development of real-time multilingual captioning systems, which will allow videos to be instantly captioned in multiple languages, breaking down language barriers globally.

Additionally, incorporating emotion and sentiment recognition will enable captions to convey not just dialogue but also the emotional tone and context of scenes, providing a richer and more immersive user experience.

Context-aware summarization is another promising enhancement, where systems will generate concise, intelligent summaries of video content, aiding in faster content consumption. Transformer-based models and attention mechanisms are expected to become even more refined, improving the accuracy, fluency, and cultural sensitivity of captions. Furthermore, the adoption of self-supervised learning will reduce dependence on manually labelled data, accelerating model training across diverse domains.

Ethical advancements such as bias mitigation, privacy preservation, and fairness in caption generation will be crucial, ensuring inclusivity and trust in AI-driven systems. Ultimately, the future of video captioning aims to create more adaptive, accessible, and intelligent systems that cater to a wide range of users and applications, fostering a universally inclusive digital environment. In addition to advancements in core technologies, future video captioning systems are expected to integrate with wearable devices, augmented reality (AR), and virtual reality (VR) platforms. This will allow captions to be overlaid seamlessly onto a user's real-world or virtual environment, enhancing accessibility for immersive experiences in gaming, education, and remote collaboration.

Moreover, adaptive learning mechanisms could be incorporated, where captioning models continually improve based on user feedback, learning preferences, and evolving language use, ensuring that captions remain accurate, user-centric, and contextually relevant over time. The ability to customize captions — adjusting font size, color, background, and placement — will empower users with varying needs to personalize their viewing experience.

Another area of future exploration is the use of multimodal AI systems that can simultaneously process audio, visual, and textual information to produce even richer, more descriptive captions, capable of narrating subtle actions, environmental sounds, or mood shifts that are often missed today. Cross-domain transfer learning could enable captioning models trained on general videos to adapt quickly to specialized fields such as medical, legal, or technical content with minimal retraining.

Finally, regulatory frameworks are likely to evolve, making accessible captioning a mandatory standard rather than an optional feature, ensuring that accessibility becomes a fundamental design principle across all digital platforms. As the demand for inclusivity grows, video captioning will continue to play a central role in shaping a truly universal, intelligent, and human-centred digital communication ecosystem.

## REFERENCES

- [1]. M. Mahrishi, S. Morwal, A. W. Muzaffar, S. Bhatia, P. Dadheech, and M. K. I. Rahmani, "YOLOv4 and custom feature extraction for the video are identified in this paper," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 2, pp. 577–590, Sept. 2020.
- [2]. M. S. Hutchinson and V. N. Gadepalli, "Understanding video action: The action recognition issue, video dataset completion, and video data preparation methods," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 3, pp. 578–589, Jan. 2020.
- [3]. X. Long, "Creation of an end-to-end deep learning framework for video generation, translation, and sign language recognition," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 2, pp. 577–589, March. 2019.
- [4]. Z. Zhou, V. W. L. Tam, and E. Y. Lam, "Sign BERT: A deep learning framework for continuous recognition of sign language based on BERT," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 8, pp. 677–689, Nov. 2021.
- [5]. Y. Tu, C. Zhou, J. Guo, S. Gao, and Z. Yu, "Video captioning with multi-faceted attention," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 8, pp. 998–1008, Nov. 2018.
- [6]. U. A. Khan, M. A. Martinez-Del-Amor, S. M. Altowaijri, A. Ahmed, A. U. Rahman, N. U. Sama, K. Haseeb, and N. Islam, "Movie tags prediction and segmentation using deep learning," *IEEE Access*, vol. 8, pp. 60716086, 2020.
- [7]. C. Huang and H. Wang, "A novel key-frames selection framework for comprehensive video summarization," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 2, pp. 577–589, Feb. 2020.
- [8]. R. Mishra, "Videoshotboundarydetection using hybrid dual tree complex wavelet transform with Walsh Hadamard transform," *Multimedia Tools Appl.*, vol. 80, no. 18, pp. 28109–28135, Jul. 2021.
- [9]. S. Chakraborty, A. Singh, and D. M. Thounaojam, "A novel bifold-stage shot boundary detection algorithm: Invariant to motion and illumination," *Vis. Comput.*, Jan. 2021.

# APPENDIX A

## CODING

### FEATURE EXTRACTION

```
import shutil
import tqdm
import numpy as np
import cv2
import os
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.models import Model
import config

def video_to_frames(video):
    path = os.path.join(config.test_path, 'temporary_images')
    if os.path.exists(path):
        shutil.rmtree(path)
    os.makedirs(path)
    video_path = os.path.join(config.test_path, 'video', video)
    count = 0
    image_list = []
    # Path to video file
    cap = cv2.VideoCapture(video_path)
    while cap.isOpened():
        ret, frame = cap.read()
        if ret is False:
            break
        cv2.imwrite(os.path.join(config.test_path, 'temporary_images', 'frame%d.jpg' % count), frame)
        image_list.append(os.path.join(config.test_path, 'temporary_images', 'frame%d.jpg' % count))
        count += 1

    cap.release()
    cv2.destroyAllWindows()
    return image_list

def model_cnn_load():
    model = VGG16(weights="imagenet", include_top=True, input_shape=(224, 224, 3))
    out = model.layers[-2].output
    model_final = Model(inputs=model.input, outputs=out)
    return model_final

def load_image(path):
    img = cv2.imread(path)
    img = cv2.resize(img, (224, 224))
```

```

return img

def extract_features(video, model):
    """
    :param video: The video whose frames are to be extracted to convert into a numpy array
    :param model: the pretrained vgg16 model
    :return: numpy array of size 4096x80
    """
    video_id = video.split(".")[0]
    print(video_id)
    print(f'Processing video {video}')

    image_list = video_to_frames(video)
    samples = np.round(np.linspace(
        0, len(image_list) - 1, 80))
    image_list = [image_list[int(sample)] for sample in samples]
    images = np.zeros((len(image_list), 224, 224, 3))
    for i in range(len(image_list)):
        img = load_image(image_list[i])
        images[i] = img
    images = np.array(images)
    fc_feats = model.predict(images, batch_size=128)
    img_feats = np.array(fc_feats)
    # cleanup
    shutil.rmtree(os.path.join(config.test_path, 'temporary_images'))
    return img_feats

def extract_feats_pretrained_cnn():
    """
    saves the numpy features from all the videos
    """
    model = model_cnn_load()
    print('Model loaded')

    if not os.path.isdir(os.path.join(config.test_path, 'feat')):
        os.mkdir(os.path.join(config.test_path, 'feat'))

    video_list = os.listdir(os.path.join(config.test_path, 'video'))

    # When running the script on Colab an item called '.ipynb_checkpoints'
    # is added to the beginning of the list causing errors later on, so the next line removes it.
    video_list.remove('.ipynb_checkpoints')

    for video in video_list:

        outfile = os.path.join(config.test_path, 'feat', video + '.npy')
        img_feats = extract_features(video, model)
        np.save(outfile, img_feats)

if __name__ == "__main__":

```



```
extract_feats_pretrained_cnn()
```

## CAPTION GENERATOR

```
import functools
import operator
import os
import cv2
import time
```

```
import numpy as np
import extract_features
```

```
import config
import model
```

```
class VideoDescriptionRealTime(object):
```

```
    """
```

```
        Initialize the parameters for the model
```

```
    """
```

```
    def __init__(self, config):
```

```
        self.latent_dim = config.latent_dim
```

```
        self.num_encoder_tokens = config.num_encoder_tokens
```

```
        self.num_decoder_tokens = config.num_decoder_tokens
```

```
        self.time_steps_encoder = config.time_steps_encoder
```

```
        self.max_probability = config.max_probability
```

```
        # models
```

```
        self.tokenizer, self.inf_encoder_model, self.inf_decoder_model = model.inference_model()
```

```
        self.inf_decoder_model = None
```

```
        self.save_model_path = config.save_model_path
```

```
        self.test_path = config.test_path
```

```
        self.search_type = config.search_type
```

```
        self.num = 0
```

```
    def greedy_search(self, loaded_array):
```

```
        """
```

```
        :param f: the loaded numpy array after creating videos to frames and extracting features
```

```
        :return: the final sentence which has been predicted greedily
```

```
        """
```

```
        inv_map = self.index_to_word()
```

```
        states_value = self.inf_encoder_model.predict(loaded_array.reshape(-1, 80, 4096))
```

```
        target_seq = np.zeros((1, 1, 1500))
```

```
        final_sentence = "
```

```
        target_seq[0, 0, self.tokenizer.word_index['bos']] = 1
```

```
        for i in range(15):
```

```
            output_tokens, h, c = self.inf_decoder_model.predict([target_seq] + states_value)
```

```
            states_value = [h, c]
```

```
            output_tokens = output_tokens.reshape(self.num_decoder_tokens)
```

```
            y_hat = np.argmax(output_tokens)
```

```
            if y_hat == 0:
```

```

        continue
    if
        inv_map[y_hat
        ] is None:
            break
    if inv_map[y_hat] == 'eos':
        break
    else:
        final_sentence = final_sentence + inv_map[y_hat] + ''
        target_seq = np.zeros((1, 1, 1500))
        target_seq[0, 0, y_hat] = 1
    return final_sentence

def decode_sequence2bs(self, input_seq):
    states_value = self.inf_encoder_model.predict(input_seq)
    target_seq = np.zeros((1, 1, self.num_decoder_tokens))
    target_seq[0, 0, self.tokenizer.word_index['bos']] = 1
    self.beam_search(target_seq, states_value, [], [], 0)
    return decode_seq

def beam_search(self, target_seq, states_value, prob, path, lens):
    """
    :param target_seq: the array that is fed into the model to predict the next word
    :param states_value: previous state that is fed into the lstm cell
    :param prob: probability of predicting a word
    :param path: list of words from each sentence
    :param lens: number of words
    :return: final sentence
    """
    global decode_seq
    node = 2
    output_tokens, h, c = self.inf_decoder_model.predict(
        [target_seq] + states_value)
    output_tokens = output_tokens.reshape(self.num_decoder_tokens)
    sampled_token_index = output_tokens.argsort()[-node:][::-1]
    states_value = [h, c]
    for i in range(node):
        if sampled_token_index[i] == 0:
            sampled_char = "
        else:
            sampled_char = list(self.tokenizer.word_index.keys())[
                list(self.tokenizer.word_index.values()).index(sampled_token_index[i])]
    MAX_LEN = 12
    if sampled_char != 'eos' and lens <= MAX_LEN:
        p = output_tokens[sampled_token_index[i]]
        if sampled_char == "
            p = 1
        prob_new = list(prob)
        prob_new.append(p)
        path_new = list(path)
        path_new.append(sampled_char)
        target_seq = np.zeros((1, 1, self.num_decoder_tokens))
        target_seq[0, 0, sampled_token_index[i]] = 1.
        self.beam_search(target_seq, states_value, prob_new, path_new, lens + 1)

```

```

else:
    p = output_tokens[sampled_token_index[i]]
    prob_new = list(prob)
    prob_new.append(p)
    p = functools.reduce(operator.mul, prob_new, 1)
    if p > self.max_probability:
        decode_seq = path
        self.max_probability = p

def decoded_sentence_tuning(self, decoded_sentence):
    # tuning sentence
    decode_str = []
    filter_string = ['bos', 'eos']
    uni_gram = {}
    last_string = ""
    for idx2, c in enumerate(decoded_sentence):
        if c in uni_gram:
            uni_gram[c] += 1
        else:
            uni_gram[c] = 1
        if last_string == c and idx2 > 0:
            continue
        if c in filter_string:
            continue
        if len(c) > 0:
            decode_str.append(c)
        if idx2 > 0:
            last_string = c
    return decode_str

def index_to_word(self):
    # inverts word tokenizer
    index_to_word = {value: key for key, value in self.tokenizer.word_index.items()}
    return index_to_word

def get_test_data(self):
    # loads the features array
    file_list = os.listdir(os.path.join(self.test_path, 'video'))
    # with open(os.path.join(self.test_path, 'testing.txt')) as testing_file:
    #     lines = testing_file.readlines()
    # file_name = lines[self.num].strip()
    file_name = file_list[self.num]
    path = os.path.join(self.test_path, 'feat', file_name + '.npy')
    if os.path.exists(path):
        f = np.load(path)
    else:
        model = extract_features.model_cnn_load()
        f = extract_features.extract_features(file_name, model)
    if self.num < len(file_list):
        self.num += 1
    else:
        self.num = 0
    return f, file_name

def test(self):

```

```

X_test, filename = self.get_test_data()
# generate inference test outputs if
self.search_type == 'greedy':
    sentence_predicted = self.greedy_search(X_test.reshape((-1, 80, 4096)))
else:
    sentence_predicted = "
    decoded_sentence = self.decode_sequence2bs(X_test.reshape((-1, 80, 4096)))
    decode_str = self.decoded_sentence_tuning(decoded_sentence)
    for d in decode_str:
        sentence_predicted = sentence_predicted + d + '
# re-init max prob
self.max_probability = -1
return sentence_predicted, filename

def main(self, filename, caption):
    """
    :param filename: the video to load
    :param caption: final caption
    :return:
    """

    # 1. Initialize reading video object
    cap1 = cv2.VideoCapture(os.path.join(self.test_path, 'video', filename))
    cap2 = cv2.VideoCapture(os.path.join(self.test_path, 'video', filename))
    caption = '[' + ' '.join(caption.split()[1:]) + ']'
    # 2. Cycle through pictures
    while cap1.isOpened():
        ret, frame = cap2.read()
        ret2, frame2 = cap1.read()
        if ret:
            imS = cv2.resize(frame, (480, 300))
            cv2.putText(imS, caption, (100, 270), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 0),
                        2, cv2.LINE_4)
            cv2.imshow("VIDEO CAPTIONING", imS)
        if ret2:
            imS = cv2.resize(frame, (480, 300))
            cv2.imshow("ORIGINAL", imS)
        else:
            break

    # Quit playing
    key = cv2.waitKey(25)
    if key == 27: # Button esc
        break

    # 3. Free resources
    cap1.release()
    cap2.release()
    cv2.destroyAllWindows()

if __name__ == "__main__":
    video_to_text = VideoDescriptionRealTime(config)
    while True:
        print(' ..... \nGenerating Caption:\n')

```

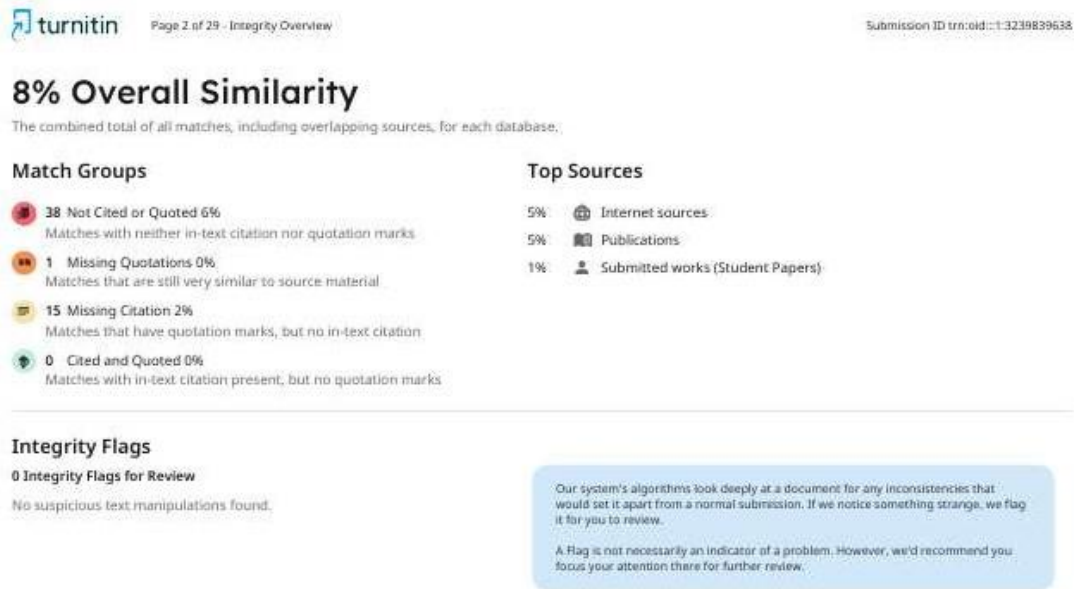
```

start = time.time()
video_caption, file = video_to_text.test()
end = time.time()
sentence = ""
print(sentence)
for text in video_caption.split():
    sentence = sentence + ' ' + text
    print("\n.....\n")
    print(sentence)
print("\n.....\n")
print('It took {:.2f} seconds to generate caption'.format(end-start))
video_to_text.main(file, sentence)
play_video = input('Should I play the video? ')
if play_video.lower() == 'y':
    continue
elif play_video.lower() == 'n':
    break
else:
    print('Could not understand type (y) for yes and (n) for no')
    continue

```

# APPENDIX B

## PLAGIARISM REPORT



### Match Groups

- **38 Not Cited or Quoted 6%**  
Matches with neither in-text citation nor quotation marks
- **1 Missing Quotations 0%**  
Matches that are still very similar to source material
- **15 Missing Citation 2%**  
Matches that have quotation marks, but no in-text citation
- **0 Cited and Quoted 0%**  
Matches with in-text citation present, but no quotation marks

### Top Sources

- 5% Internet sources
- 5% Publications
- 1% Submitted works (Student Papers)

### Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

<b>1</b>	Internet	github.com	1%
<b>2</b>	Publication	S. Durga, S. Deepakanmani, Esther Daniel, "2D convolution neural network-based...	<1%
<b>3</b>	Internet	engrxiv.org	<1%
<b>4</b>	Internet	www.techscience.com	<1%
<b>5</b>	Internet	thebusinessresearchcompany.com	<1%
<b>6</b>	Publication	Yash Mishra, Kedarnath Senapati. "Deep Learning-based Image Caption Generat...	<1%
<b>7</b>	Student papers	EARLY MAKERS Group SA	<1%
<b>8</b>	Publication	"Pattern Recognition and Computer Vision", Springer Science and Business Media...	<1%
<b>9</b>	Internet	ksascholar.dri.sa	<1%
<b>10</b>	Publication	Vivek S. Sharma, Shubham Mahajan, Anand Nayyar, Amit Kant Pandit. "Deep Lear...	<1%

11	Student papers	Debre Berhan University	<1%
12	Student papers	Mepco Schlenk Engineering college	<1%
13	Student papers	University of Wales Institute, Cardiff	<1%
14	Internet	www.dice.com	<1%
15	Internet	financialit.net	<1%
16	Internet	oup.com.pk	<1%
17	Internet	rsisinternational.org	<1%
18	Internet	www.ai-hive.net	<1%
19	Internet	www.mdpi.com	<1%
20	Internet	academic-accelerator.com	<1%
21	Internet	bpasjournals.com	<1%
22	Internet	conrel.sice.umkc.edu	<1%
23	Publication	"Natural Language Processing and Chinese Computing", Springer Science and Bu...	<1%
24	Publication	Emmanuel Kwaku Ofori, Seth Kwabena Amponsah, Yashwant V. Pathak. "Diagnos...	<1%




25	Publication	MD. Zakir Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, Hamid Laga. "A Com...	<1%
26	Publication	Momiao Xiong. "Artificial Intelligence and Causal Inference", CRC Press, 2022	<1%
27	Internet	datahorizonresearch.com	<1%
28	Internet	staging-medinform.jmir.org	<1%
29	Internet	www.ghanamedicalvolunteers.org	<1%
30	Internet	www.researchgate.net	<1%
31	Publication	Wangyu Choi, Jiasi Chen, Jongwon Yoon. "PWS-DVC: Enhancing Weakly Supervised...	<1%

## APPENDIX C

### CONFERENCE PROOF

[ICCSP 2025] #1571135924 has been uploaded External Inbox

 **ICCSP 2025** <iccsp2025-chairs@edas.info> Mon, Apr 7, 2:45 AM ★ ↶ ⋮


to Vasu, 5, me

Dear Mr. Vasu Rastogi:

Thank you for uploading your paper for paper 1571135924 (AI-Enhanced Video Captioning Using VGG16 & CNN) to 2025 11th International Conference on Communication and Signal Processing (ICCSP). The paper is of type application/pdf and has a length of 359189 bytes.

You can modify your paper at [1571135924](https://edas.info/index.php?c=32604) and see all your submissions at <https://edas.info/index.php?c=32604> using the EDAS identifier [va7844@smst.edu.in](mailto:va7844@smst.edu.in)

Regards,  
The Conference Chairs  
ICCSP 2025  
Adhiparasakthi Engineering College  
Meimaruvaithur - 603319  
Chengalpattu District  
Tamilnadu  
India  
email: [iccsp@apecc.edu.in](mailto:iccsp@apecc.edu.in)  
website: <https://iccp.apecc.edu.in/>

 **VASU RASTOGI (RA2111026010295)** 2:11PM (3 minutes ago) ★ ↶ ⋮

to me

----- Forwarded message -----  
From: **ICCSP 2025** <iccsp2025-chairs@edas.info>  
Date: Mon, 7 Apr 2025 at 2:48 AM  
Subject: [ICCSP 2025] #1571135924 has been uploaded  
To: Vasu Rastogi <[va7844@smst.edu.in](mailto:va7844@smst.edu.in)>, S Sadagopan <[sadagopan@smst.edu.in](mailto:sadagopan@smst.edu.in)>, Arushi Pataniyakk <[arushi119@smst.edu.in](mailto:arushi119@smst.edu.in)>

Dear Mr. Vasu Rastogi:

Thank you for uploading your paper for paper 1571135924 (AI-Enhanced Video Captioning Using VGG16 & CNN) to 2025 11th International Conference on Communication and Signal Processing (ICCSP). The paper is of type application/pdf and has a length of 359189 bytes.

You can modify your paper at [1571135924](https://edas.info/index.php?c=32604) and see all your submissions at <https://edas.info/index.php?c=32604> using the EDAS identifier [va7844@smst.edu.in](mailto:va7844@smst.edu.in)

Regards,  
The Conference Chairs

ICCSP 2025  
Adhiparasakthi Engineering College  
Meimaruvaithur - 603319  
Chengalpattu District  
Tamilnadu  
India