Video Captioning using Keras

A **Project report** submitted in partial fulfillment of the

Requirements for the degree of

Bachelor of Technology

By

Amol Pawar	**ET-B 24**
Sawan Damase	**ET-A 25**
Shubham Potphode	**ET-B 33**
Rushikesh Ugalmugale	**ET-B 65**

Under the guidance of

Prof.Suhas Bhise

DEPARTMENT OF

ELECTRONICS & TELECOMMUNICATION ENGINEERING

VISHWAKARMA INSTITUTE OF TECHNOLOGY PUNE

2020-21

Bansilal Ramnath Agarwal Charitable Trust's

VISHWAKARMA INSTITUTE OF TECHNOLOGY, PUNE – 37

(An Autonomous Institute Affiliated to Savitribai Phule Pune University)

CERTIFICATE

This is to certify that the **Project Report** entitled **Video Captioning using Keras** has been submitted in the academic year **2020-21** by

Amol Pawar	**ET-B 24**
Sawan Damase	**ET-A 25**
Shubham Potphode	**ET-B 33**
Rushikesh Ugalmugale	**ET-B 65**

Under the supervision of **Prof.Suhas Bhise** in partial fulfillment of the requirements for the degree of Bachelor of Technology in **Electronics and Telecommunication Engineering** as prescribed by Savitribai Phule Pune University.

Guide/Supervisor

Name: Prof.Suhas Bhise

Signature:

Head of the Department

Name: Prof. Dr. Shripad Bhatlawande

Signature:

External Examiner

Name:

Signature:

Acknowledgments

On the very outset of this report, we would like to extend our sincere and heartfelt obligation towards all the personages who have helped us in this endeavor. Without their active guidance, help, co-operation and encouragement, we would not have made a headway in the project and report preparation.

We pay our deep sense of gratitude to our project guide, Prof..Suhas Bhise for his guidance and support starting right from the literature survey of the project to the final implementation and interpretation of results. We are immensely obliged to him for his elevating inspiration, encouraging guidance, and kind supervision in the completion of our project.

Date: 22/01/2022

Amol Pawar	**ET-B 24**
Sawan Damase	**ET-A 25**
Shubham Potphode	**ET-B 33**
Rushikesh Ugalmugale	**ET-B 65**

Abstract

Video captioning is the process of collecting information from a video and using those attributes to create video captions. The computer vision field faces a major difficulty in automatically generating natural language descriptions of videos. The use of 2-D and/or 3-D Convolutional Neural Networks (CNNs) to encode video material and Recurrent Neural Networks (RNNs) to decode a text has yielded the most recent development in this field. Long Short-Term Memory with Transferred Semantic Characteristics (LSTM) is a revolutionary deep architecture that integrates transferred semantic attributes learnt from images and videos into the CNN + RNN framework by training them end-to-end in this article.

Table of Contents

List of Figures

CHAPTER 1
INTRODUCTION

1.1.OVERVIEW

Video captioning, that is understood as describing videos with tongue, has brought a profound challenge to each pc vision and language process communities. Intensive analysis interests are acquired this rising topic. the event of computer science in varied fields, like pc vision, tongue process, and machine learning, has light-emitting diode to a growing interest in complicated intelligence issues that need the coincident process of tongue and pictures. With the exponential rise within the generation of video knowledge, light-emitting diode by the popularization of on-line video sharing platforms like YouTube, Dailymotion, and Netflix, analysis interest in automatic analysis of video has grown up. Typical video-based complicated intelligence issues embrace video captioning and video question-answering. during this study, we tend to decide to augment the present analysis on video captioning strategies. The sequence learning technique, in distinction, is to leverage sequence learning models to directly translate the video content into a sentence, that is especially galvanized by the recent advances by exploitation repeated Neural Networks (RNNs) in MT. The spirit behind is Associate in Nursing encoder-decoder mechanism for translation. additional specifically, Associate in Nursing encoder 2-D/3-D Convolutional Neural Network (CNN) reads a video and produces a vector of video representations, that successively is fed into a decoder RNN that generate a natural sentence. whereas encouraging performances square measure reported, the CNNs and RNNs-based sequence learning approaches translate directly from video representations to language, going away the high-level linguistics cues within the video beneath explored. Moreover, high-level linguistics info, i.e., linguistics attributes, has shown effectiveness within the vision of language tasks. Captions square measure notably useful to persons look videos in their non-native language, kids and adults learning to scan, and persons UN agency square measure D/deaf or exhausting of hearing. after you create your content accessible, you permit folks that can be deaf or exhausting of hearing to possess access to the videos you manufacture. Search engines can’t crawl video, however they'll crawl text. If you would like your videos to rank on Google, back your video up with captions and a transcript of the audio.

1.2.MOTIVATION

Hard training statistics, not visual notions, are used to develop current approaches for creating attention weights. If training movies frequently finish with humans failing, for example, the model may learn to correlate the end of the video with the words "falling down." The attention model would function as expected if the video and caption were "lady frightened man skating and he falls down." The trained attention parameters would perform poorly if the caption was "guy falls down and then does a terrific skateboard trick," since the model would seek attention over the wrong temporal region in the video. Our model extracts frame-by-frame temporal concepts across the length of the video to guide the attention mechanism to the correct place in the video. This allows the model to link certain phrases like woman, man, and skating to specific geographic places throughout the movie. Attention models weight some parts high and others low in a movie to direct attention to appropriate spatial-temporal places. By using a convex mixture of each, the model learns how to linearly merge various parts of a movie. Because weights are learned parameters that must be fixed at train time, attention models are restricted to have an equal number of regions in all samples. We introduce parametric Gaussian attention models that learn a continuous function and sample it at each discrete region to make the attention mechanism independent of video duration.

1.3.SCOPE OF THE PROJECT

In this project Video captioning system is implemented using keras and LSTM. MSVD dataset is used to train and test.

1.4.METHODOLOGIES OF PROBLEM SOLVING

To achieve the different problems such as Video captioning following methodologies are used.

- Keras
- LSTM
- VGG16

1.5.OBJECTIVES

- VGG 16 used which is based on Convolutional Neural network.

CHAPTER 2
LITERATURE SURVEY

So as of now, there are a lot of methods proposed for Video captioning with Keras. So we will discuss existing systems in brief, first. There are many methods for Video captioning with Keras. Generally, they are LSTM architecture by using deep learning-based, Machine learning-based.

To accurately represent a video, the suggested video captioning model uses both visual and semantic characteristics. Visual features are extracted using the existing ResNet (residual network) and C3D (3DConvNet) convolutional neural networks (CNN), while semantic features are extracted using novel semantic feature networks. Semantic features are divided into dynamic features. expressing video motions as well as static semantic characteristics such as objects, people, and the background[3][1].

The encoder-decoder architecture for video captioning is made up of two parts: one extracts information from a video using an encoder, and the other uses a decoder to output caption word sequences. CNN models such as pre-trained very deep convolutional networks (VGG), ResNet, and C3D, in particular, have been employed as encoders, whereas recurrent neural networks (RNN) have primarily been used as decoders.[1][4].

The author looks into how linguistic data gleaned from massive text corporations can help with the creation of natural language video descriptions. Specifically, They integrate both a neural language model and distributional semantics trained on large text corpora into a recent LSTM-based architecture for video description. We evaluate our approach on a collection of Youtube videos as well as two large movie description datasets showing significant improvements in grammaticality while modestly improving descriptive quality[2].

Long Short-Term Memory with Transferred Semantic Attributes (LSTM-TSA) is a novel deep architecture that incorporates the transferred semantic attributes learned from images and videos into the CNN plus RNN framework, by training them in an end-to-end manner.The notion that semantic qualities play a key role in captioning and that images and videos have complimentary semantics and hence can reinforce each other for captioning motivated the invention of LSTM-TSA.[1][3] To boost video captioning, the Author proposes a novel transfer unit to model the mutually correlated attributes learn from images and videos.

LSTM-TSA achieves to-date the best-published performance in sentence generation on MSVD: 52.8% and 74.0% in terms of BLEU@4 and CIDEr-D.[3] Superior results are also reported on M-VAD and MPII-MD when compared to state-of-the-art methods. propose a novel data-set that contains transcribed ADs, which are temporally aligned to full-length HD movies.[4] In addition, the author also collected the aligned movie scripts which have been used in prior work and compared the two different sources of descriptions. The MPII Movie Description data set (MPII-MD) includes a parallel corpus of over 68K phrases and video samples from 94 HD movies in total. We characterize the data set by benchmarking different approaches for generating video descriptions [5]. We propose to produce textual subtitles for the provided input video using a neural network-based framework.[6]. Deep learning approaches, especially deep Con-evolution Neural Networks (ConvNets), have achieved overwhelming accuracy with fast processing speed for image classification. [8]. The state-of-the-art on video captioning benchmarks. HRNE outperforms all contemporary systems that combine multiple inputs, such as RGB ConvNet plus 3D ConvNet, even when employing a single network with only RGB stream as input.

The first stage is feature extraction, in which we extract features from the input video that are both whole video and keyframe image-based. As the whole video-based feature we use dense trajectories Keyframe image features are extracted by feeding these images through Convolutional Neural Networks (CNN) trained on the ImageNet database [8].

CHAPTER 3
SOFTWARE REQUIREMENT SPECIFICATION

3.1 Software Specification

3.1.1. Language: Python

Python is featured with a dynamic type system, automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It has a large and comprehensive standard library.

Python is an interpreter, object-oriented, high-level programming language with dynamic semantics. It's high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development. Python is simple, easy to learn and syntax emphasizes readability, therefore, it reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms and can be freely distributed.

3.1.2. Image processing Library: Open CV 4.5

OpenCV is an image processing library created by Intel and it is maintained by the Willow Garage after 2007. It is available for C, C++, and Python. It is Open source and easily available on the website "WWW.opencv.willowgarage.com/wiki/". You just download it and use it. This site is updated yearly and the version is also launched yearly. It is easy to handle because all functions are available within a library.

Open CV (Free Open-Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. It has a BSD license (free for commercial or research use). Open CV was originally written in C but now has a full C++ interface and all new development is in C++. There is also a full Python interface to the library.

Open-Source Computer Vision Library, also known as Open CV, is an open-source freeware which is aimed at computer vision. Among its applications include object identification and motion tracking. It is used in this project because of its versatility as well as the fact that it has a C++ interface. Open CV runs on most of the major Operating Systems (OS), which makes it useful when using another computer to program or test.

3.1.3. Anaconda

Anaconda is a premium open-source distribution of the Python and R programming languages for large-scale data processing, predictive analytic, and scientific computing, that aims to simplify package management and deployment Package versions are managed by the package management system conda.

3.1.4. Algorithm

- Beam Search
- VGG-16

3.1.5. Spyder

Spyder is a powerful environment built in Python, for Python, and designed by and for scientists, engineers, and data analysts. It offers a unique amalgamation of the advanced editing, analysis, debugging, and profiling functionality of a wide-ranging development tool with the data study, interactive implementation, deep examination, and beautiful visualization capabilities of a scientific package.

Beyond its many built-in features, its abilities can be extensive even further via its plug-in system and API. Also, Spyder can be used as a PyQt5 extension library, allowing developers to develop upon its functionality and embed it's the mechanism, such as the interactive console, in their PyQt software.

- **Tensorflow**

TensorFlow is an end-to-end open-source environment for ML. TensorFlow is a wealthy system to handle all aspects of the ML system. However, this class focuses on using a particular TensorFlow API to develop and train ML models.

TensorFlow APIs are set hierarchically. ML researchers use low-level APIs to create and explore new ML algorithms. In this class, you will use a high-level API named tf.keras to define and train ML models and to make predictions. tf.keras is the TensorFlow variant of the open-source Keras API.

TensorFlow is a second-generation system by Google Brains. The first version was released on 11th February 2017. The implementation runs on single devices, TensorFlow can execute on multiple CPUs and GPUs. TensorFlow is present on 64 bit Linux, macOS, Windows, and mobile computing platforms together with Android and iOS. It is available for server clusters and mobile devices.

- **Keras**

Keras is an open-source NN library written in Python. It can be executed on top of Tensorflow Microsoft Cognitive Toolkit, R-language, Theano, or PlaidML. It is developed to enable superior experimentation with DNN, it focuses on being user friendly, modular, and extensible. It was developed as part of the study effort of project ONEIROS, and it's primary author and maintainer are Francois Chollet, a Google engineer.

Keras was created to be user-friendly, modular, simple to enlarge, and to work with Python. The API was "designed for human beings, not machines", and "follows best practices for reducing cognitive load".

Neural layers, cost functions, optimizers, initialization schemes, activation functions, and regularization schemes are all standalone modules that you can unite to create new models. New modules are simple to add, as new classes and functions. Models are defined in Python code, not separate model configuration files.

Keras properly doesn't do its low-level operations, such as tensor products and convolutions; it relies on a back-end engine. Even though Keras supports numerous back-end engines, its primary back end is TensorFlow, and its primary supporter is Google. The Keras API comes packaged in TensorFlow as tf.keras, which as mentioned previously will become the primary TensorFlow API as of TensorFlow 2.4.

3.2 Flowchart

3.2.1 Propose method

The flowchart for the proposed system is shown in Figure3.1. Data collection is the first step, followed by feature extraction from video frames, cleaning, and preprocessing. We added the bos> and eos> tokens before and after each caption as the only text preparation we did.
The model knows to start forecasting from here since bos> marks the beginning of the sentence, and eos> denotes the end of the statement, which is where the model knows to stop predicting. Most text generation problems are solved using an encoder-decoder design. What is a video, exactly? Isn't it possible to call it a sequence of images? For everything involving sequence, we always utilize RNNs or LSTMs. In our instance, an LSTM will be used.Loading data into

the model now that we know the model is also a crucial element of the training. The model was trained for 150 epochs. One epoch's training takes roughly 40 seconds to complete. For Tesla T4 training, I utilised the free version of Colab.Then loading the dataset and at last we get captions.

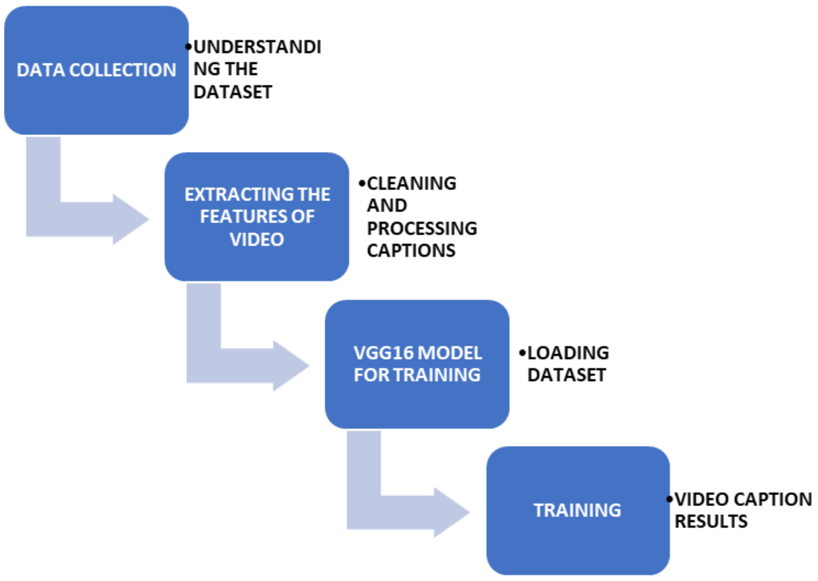
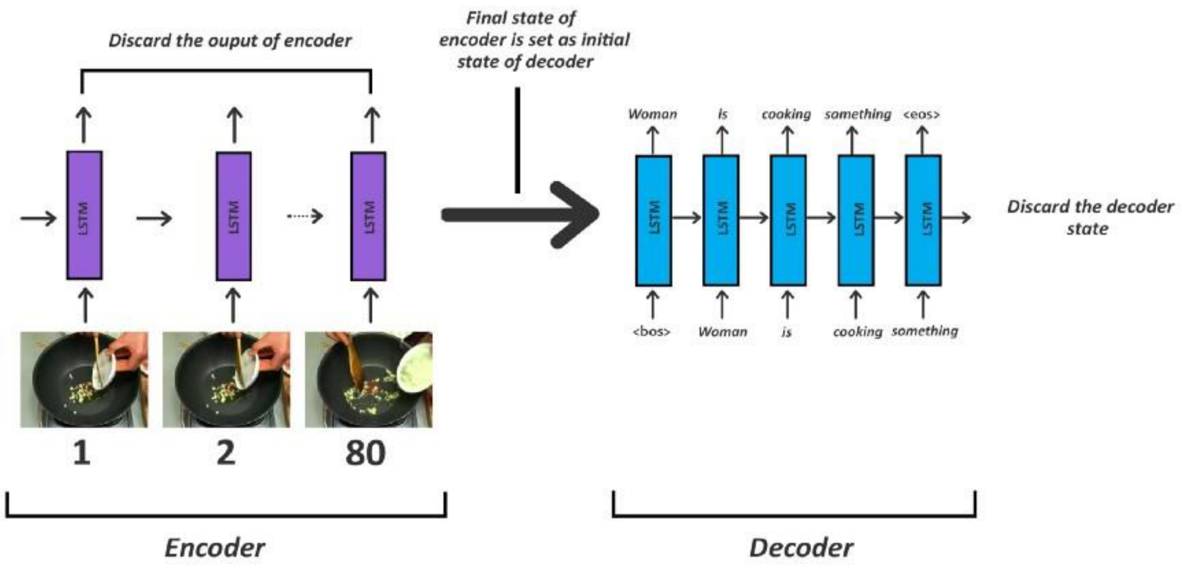

DATA COLLECTION
•UNDERSTANDING THE DATASET
EXTRACTING THE FEATURES OF VIDEO
•CLEANING AND PROCESSING CAPTIONS
VGG16 MODEL FOR TRAINING
•LOADING DATASET
TRAINING
•VIDEO CAPTION RESULTS

Fig.3.1 Flowchart of the proposed system

Encoder Decoder of LSTM network


Discard the ouput of encoder
Final state of encoder is set as initial state of decoder
LSTM
1
2
80
Encoder
Woman
is
cooking
something
<eos>
<bos>
Decoder
Discard the decoder state

3.3 METHODOLOGY

A. Dataset:

We used Microsoft's MSVD data set for this project. This data set includes 1450 short YouTube clips for training and 100 films for testing that have been hand labelled.
Each video has a unique ID, and each ID includes approximately 15–20 captions.

B. Understanding the Dataset:

The training data and testing data folders are located in the data set. Each folder has a video subfolder that contains the videos that will be used for both training and testing. There is also a feat subfolder in these folders, which stands for features. The video's features are contained in the feat folders. There are also json files for training label and testing label. The captions for each ID are contained in these json files. The following is how we can read the json files:

```
train_path='training_data'
TRAIN_LABEL_PATH=os.path.join(train_path, 'training_label.json')
# mentioning the train test split
train_split = 0.85
# loading the json file for training
with open(TRAIN_LABEL_PATH) as data_file:
    y_data = json.load(data_file)
```

C. **Extracting Features of Video:**

The number of frames taken will vary depending on the duration of the video. As a result, only 80 frames from each video are taken for the sake of simplicity. Each of the 80 frames is run through a VGG16 that has been pre-trained, and 4096 features are retrieved. The (80, 4096) shaped array is formed by stacking these characteristics. The number of frames is 80, and the number of features taken from each frame is 4096.

D. Captions Cleaning and Preprocessing:

Now we'll load all of the captions and pair them with their respective video IDs. Here's what I came up with. The train list contains two captions as well as the video ID. I added the <bos> and eos> tokens before and after each caption as the only text preprocessing I did.
The model knows to start forecasting from here since < bos> marks the beginning of the

sentence, and <eos> denotes the end of the statement, which is where the model knows to stop predicting.

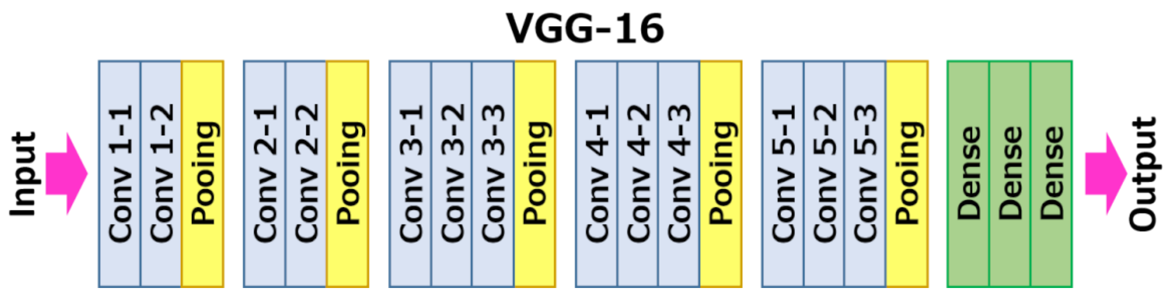
E. Model For Training:

The architecture of vgg16 model have 13 convolutional layers and 2 Fully connected layers and 1 SoftMax classifier. VGG-16 architecture has Very Deep Convolutional Network for Large Scale Image Recognition. VGG-16 is 16-layer network comprised of convolutional and fully connected layers. For simplicity, 3×3 convolutional layers were placed on top of each other.

VGG-16 has already been trained on ImageNet, which has a variety of image categories. These models are created from the scratch and trained on millions of images divided into thousands of categories using powerful GPUs. The model has learned a decent representation of low-level properties such as spatial, edges, rotation, illumination, and forms because it was trained on a large dataset. Although these additional images may belong to completely different categories than the source dataset, the pretrained model should still be able to extract important characteristics from them using transfer learning techniques.

F. VGG-16

In their publication "Very Deep Convolutional Networks for Large-Scale Image Recognition," K.Simonyan and A. Zisserman from the University of Oxford proposed the VGG16 convolutional neural network model. In ImageNet, a dataset of over 14 million images belonging to 1000 classes, the model achieves 92.7 percent top-5 test accuracy. It was one of the most well-known models submitted to the 2014 ILSVRC. It outperforms AlexNet by sequentially replacing big kernel-size filters (11 and 5 in the first and second convolutional layers, respectively) with numerous 33 kernel-size filters. The NVIDIA Titan Black GPUs were used to train VGG16 for weeks.



VGG16 Architecture

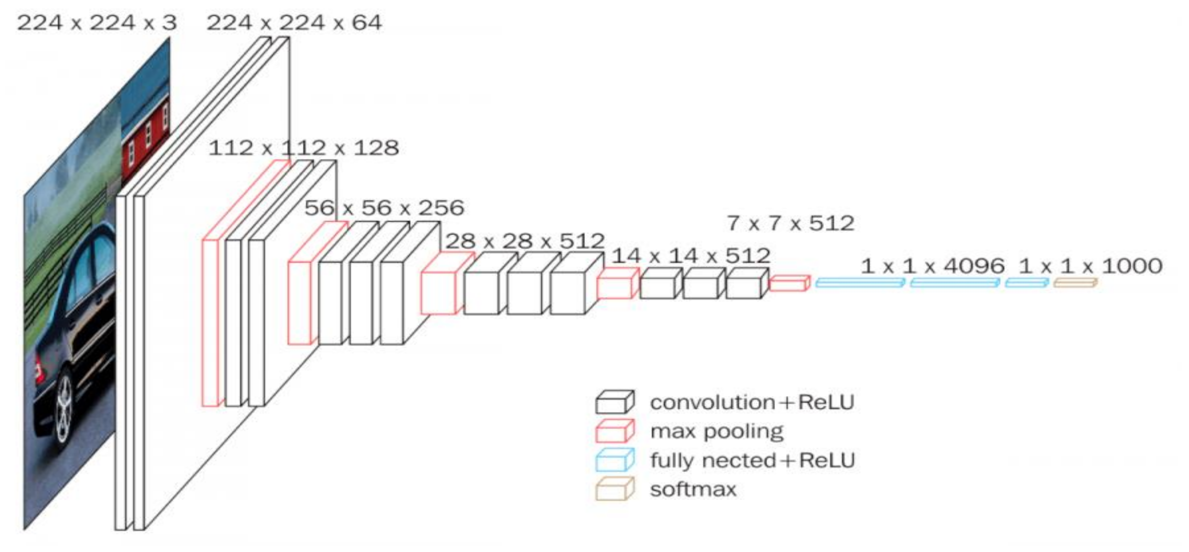


Fig: - VGG16 architecture

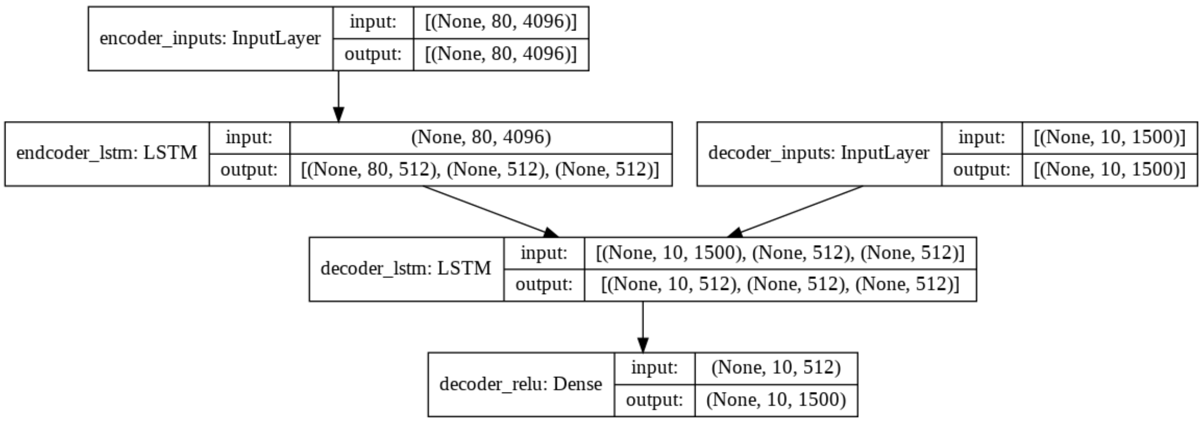
The input to the cov1 layer is a 224 by 224 RGB picture with a fixed size. The image is passed through a stack of convolutional (conv.) layers with an extremely narrow receptive field: 33 (the smallest size to capture the notions of left/right, up/down, and centre). It also uses 11 convolution filters in one of the setups, which may be thought of as a linear transformation of the input channels (followed by non-linearity). The convolution stride is set to 1 pixel, while the convolution spatial padding is set to 0. The spatial resolution of the layer input is kept after convolution, i.e. the padding is 1-pixel for 33% conv. layers. Five max-pooling layers do the spatial pooling, and they follow some of the rules.Max-pooling is done with stride 2 over a 22 pixel window.

Following a stack of convolutional layers (of varying depth in different architectures), three Fully-Connected (FC) layers are added: the first two have 4096 channels apiece, while the third performs 1000-way ILSVRC classification and hence has 1000 channels (one for each class). The soft-max layer is the final layer. In all networks, the completely connected levels are configured in the same way.
The rectification (ReLU) non-linearity is present in all hidden layers. It should also be highlighted that, with the exception of one, none of the networks incorporate Local Response Normalization (LRN), which does not improve performance on the ILSVRC dataset but increases memory consumption and computation time.

CHAPTER 4
SYSTEM ARCHITECTURE

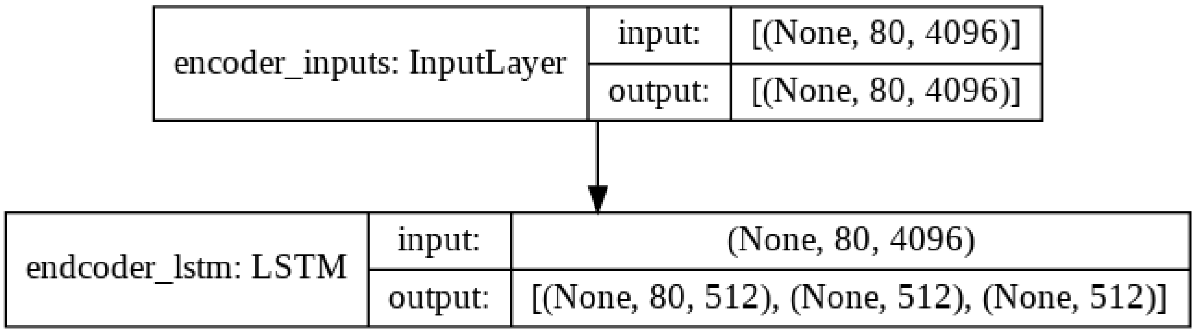
4.1.SYSTEM ARCHITECTURE

Training architecture



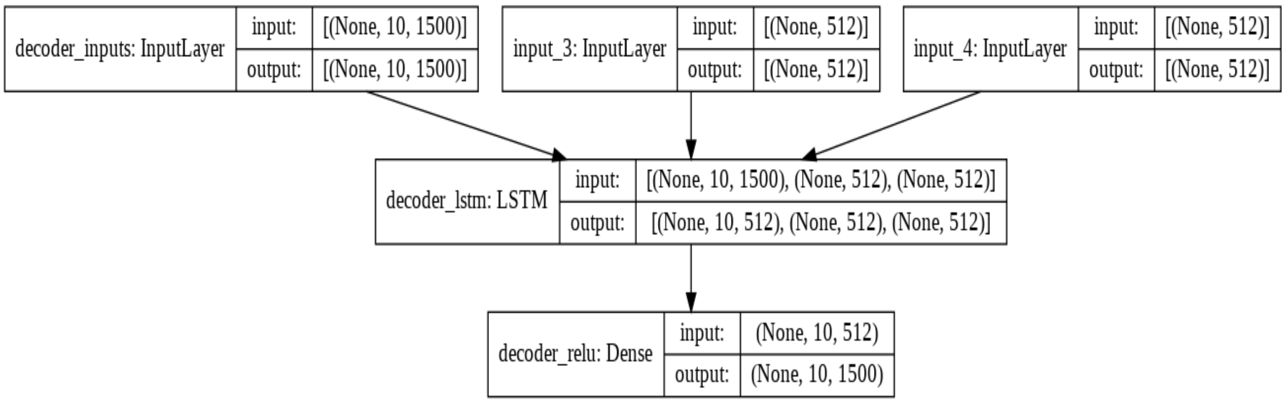
encoder_inputs: InputLayer | input: [(None, 80, 4096)] | output: [(None, 80, 4096)]
endcoder_lstm: LSTM | input: (None, 80, 4096) | output: [(None, 80, 512), (None, 512), (None, 512)]
decoder_inputs: InputLayer | input: [(None, 10, 1500)] | output: [(None, 10, 1500)]
decoder_lstm: LSTM | input: [(None, 10, 1500), (None, 512), (None, 512)] | output: [(None, 10, 512), (None, 512), (None, 512)]
decoder_relu: Dense | input: (None, 10, 512) | output: (None, 10, 1500)

Encoder



encoder_inputs: InputLayer | input: [(None, 80, 4096)] | output: [(None, 80, 4096)]
endcoder_lstm: LSTM | input: (None, 80, 4096) | output: [(None, 80, 512), (None, 512), (None, 512)]

Decoder



decoder_inputs: InputLayer | input: [(None, 10, 1500)] | output: [(None, 10, 1500)]
input_3: InputLayer | input: [(None, 512)] | output: [(None, 512)]
input_4: InputLayer | input: [(None, 512)] | output: [(None, 512)]
decoder_lstm: LSTM | input: [(None, 10, 1500), (None, 512), (None, 512)] | output: [(None, 10, 512), (None, 512), (None, 512)]
decoder_relu: Dense | input: (None, 10, 512) | output: (None, 10, 1500)

Fig.4.1 Block diagram Video captioning system

4.1.1.Input Dataset

For this project, we have used the MSVD knowledge set by Microsoft. you'll get the information about dataset from kaggle. This knowledge set contains 1450 short YouTube clips that are manually labeled for coaching and one hundred videos for testing.
Each video has been appointed a novel ID and every ID has regarding 15–20 captions.On downloading the information set you may realize the training_data and testing_data folders. every of the folders contain a video sub folder that contains the videos which will be used for coaching also as testing. These folders additionally contain a effort sub folder that is brief for options. The effort folders contain the options of the video. there's additionally a training_label and testing_label json files. These json files contain the captions for every ID

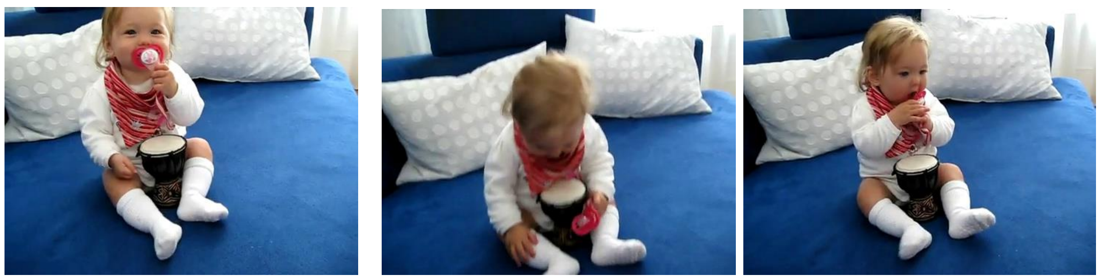


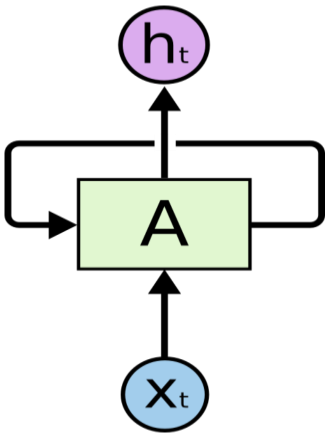
Fig 4.1.2: - Frame captures in video.

4.1.2Understanding of LSTM

A) Recurrent Neural Networks

Humans do not start thinking all over again every second. You comprehend each word as you read this essay based on your comprehension of prior words. You don't toss everything out and start again from the beginning. Your thoughts are tenacious.

This is something that traditional neural networks can't do, and it appears to be a fundamental flaw. Consider the following scenario: you wish to categorise the type of event that occurs at each moment in a movie. It's unclear how a typical neural network could use prior events in the movie to inform subsequent ones.

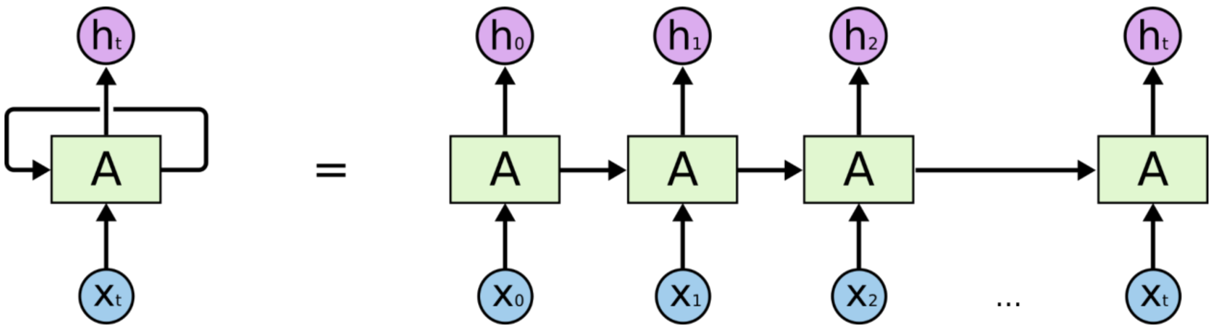


This problem is addressed by recurrent neural networks. They're networks with loops in them that allow data to endure.

Recurrent Neural Networks have loops.

A slice of neural network, AA, in the diagram above, looks at some input xtxt and outputs a value htht. Information can be transmitted from one network step to the next via a loop.

Recurrent neural networks appear enigmatic because of these loops. However, if you think about it, they're not all that different from a traditional neural network. A recurrent neural network is made up of several copies of the same network, each of which sends a message to its successor. Consider what happens if the loop is unrolled:



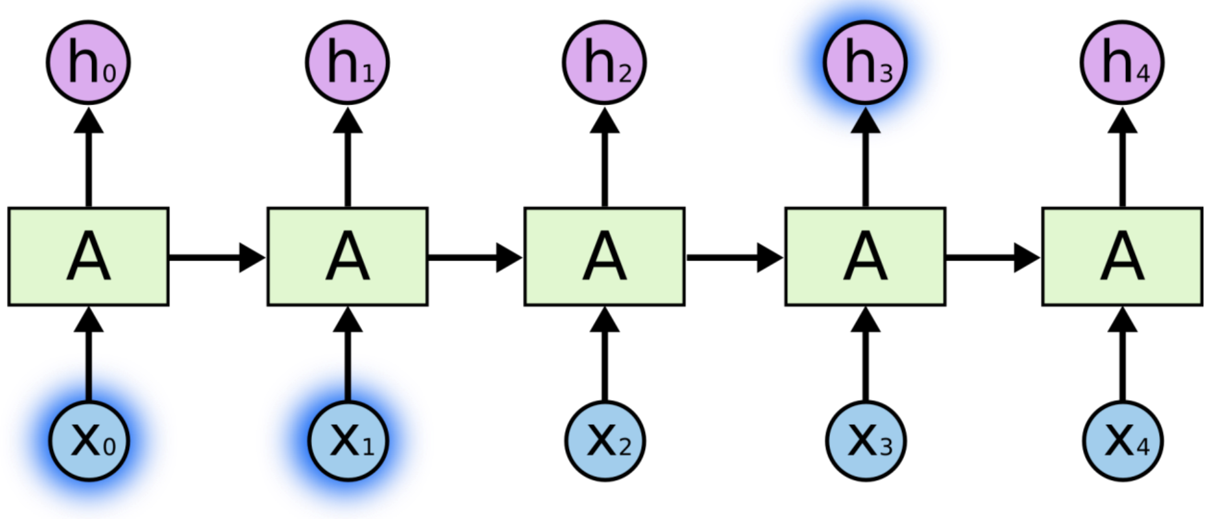
An unrolled recurrent neural network.

Recurrent neural networks are intricately tied to sequences and lists, as seen by their chain-like character. They're the most natural neural network architecture to employ for such data. And they're surely put to good use! RNNs have had remarkable success in the previous several years when applied to a range of tasks, including speech recognition, language modelling, translation, image captioning, and so on. The list could go on and on. I'll defer to Andrej Karpathy's outstanding blog piece, The Unreasonable Effectiveness of Recurrent Neural Networks, for a description of the incredible feats that RNNs can do. They are, nevertheless, very remarkable.The usage of "LSTMs," a particularly specific type of recurrent neural network that performs far better than the regular version for many tasks, is critical to these accomplishments. Almost all exciting outcomes are founded on recurrent neural networks are achieved with them.

The Problem of Long-Term Dependencies

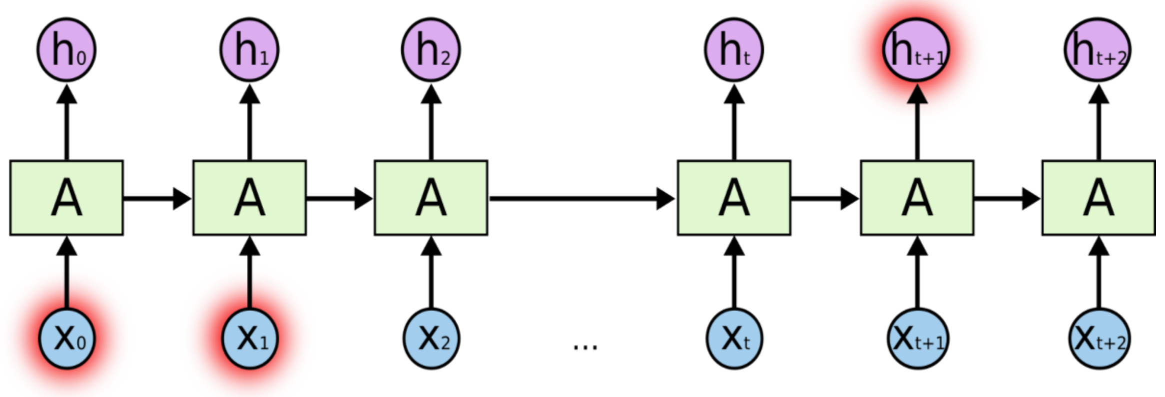
One of the allures of RNNs is the possibility of connecting earlier data to the current task, such as using previous video frames to inform comprehension of the current frame. RNNs would be immensely handy if they could accomplish this. Can they, however, do it? It is debatable.

Sometimes all we need to do is glance at recent data to complete the task at hand. Consider a language model that tries to anticipate the next word based on the ones that came before it. We don't need much more information to guess the last word in "the clouds are in the sky" — it's very evident that the following word is going to be sky. In such circumstances, where there is a discrepancy between the necessary data and the information available.



However, there are times when we require further context. Consider predicting the text's final word: "I grew up in France... I speak fluent French." According to recent evidence, the next term is most likely the name of a language, but we need the context of France from further back to narrow down which language. It's entirely feasible for the distance between relevant data and the point at which it's required to grow significantly.

Unfortunately, as the gap widens, RNNs lose their ability to learn to connect the dots.

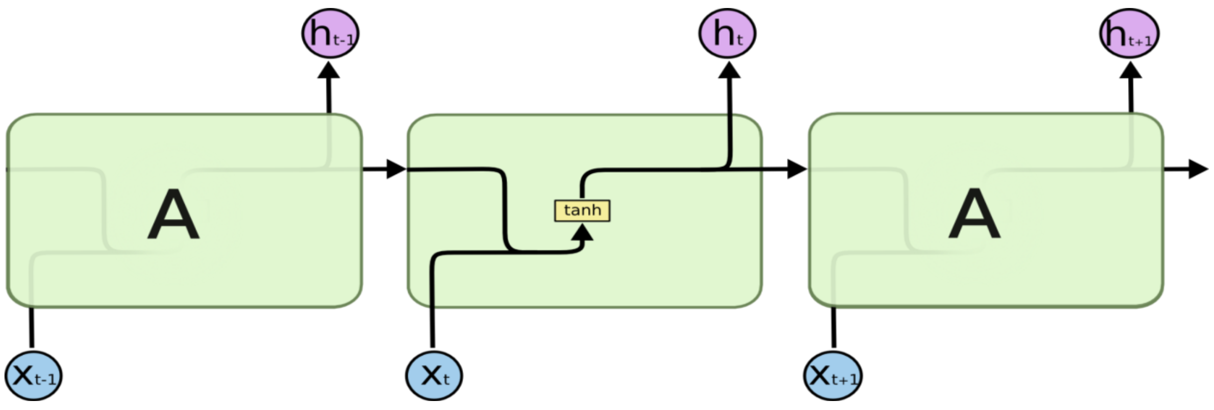


B) LSTM Networks

Long Short-Term Memory networks, or "LSTMs," are a type of RNN that can learn long-term dependencies. They are currently frequently utilized and function exceptionally effectively on a wide range of situations.

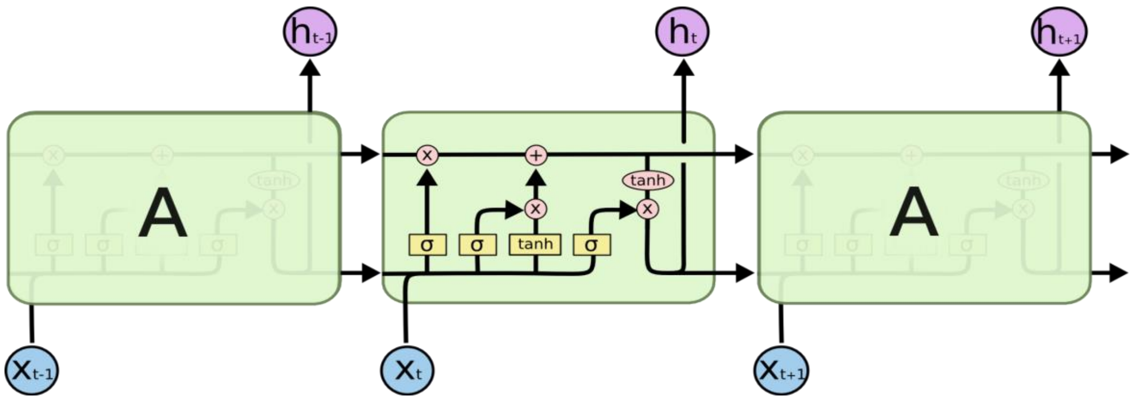
LSTMs are specifically developed to prevent the problem of long-term dependency. They don't have to work hard to remember knowledge for lengthy periods of time; it's like second nature to them!

All recurrent neural networks are made up of a series of repeated neural network modules. This repeating module in ordinary RNNs will have a relatively simple structure, such as a single tanh layer.



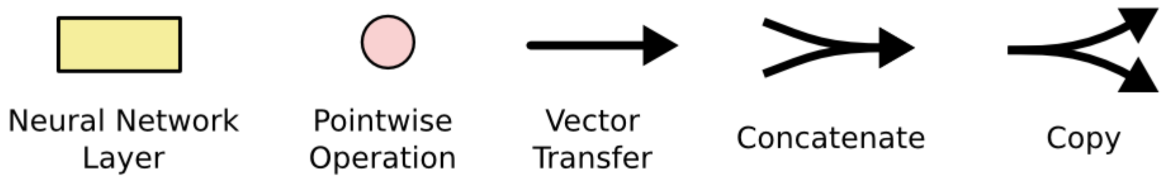
The repeating module in a standard RNN contains a single layer.

LSTMs have a chain-like structure as well, but the repeating module is different. Instead of a single neural network layer, there are four, each of which interacts in a unique way.



The repeating module in an LSTM contains four interacting layers.

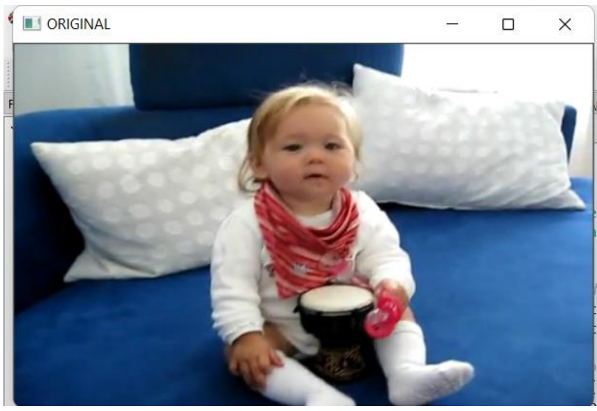
Don't be concerned with the specifics of what's going on. Later, we'll go over the LSTM diagram step by step. For the time being, let's just try to get used to the notation we'll be utilizing.

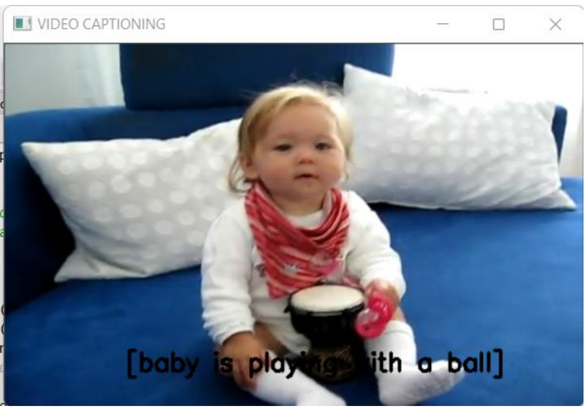


Each line in the figure above transmits a full vector from one node's output to the inputs of others. The pink circles denote pointwise operations, such as vector addition, and the yellow boxes denote learnt neural network layers. Concatenation occurs when lines merge, whereas forking occurs when a line's content is replicated and the copies are sent to various locations.

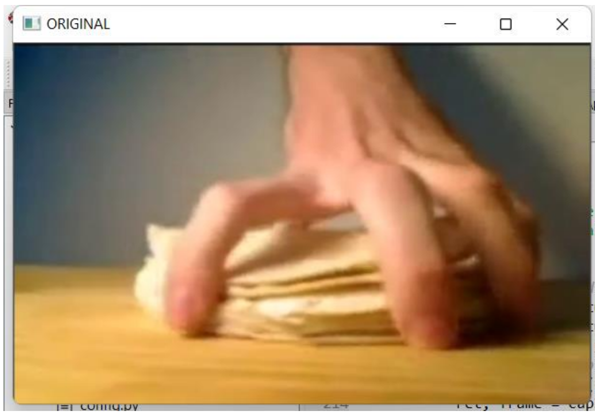
CHAPTER 5
RESULTS

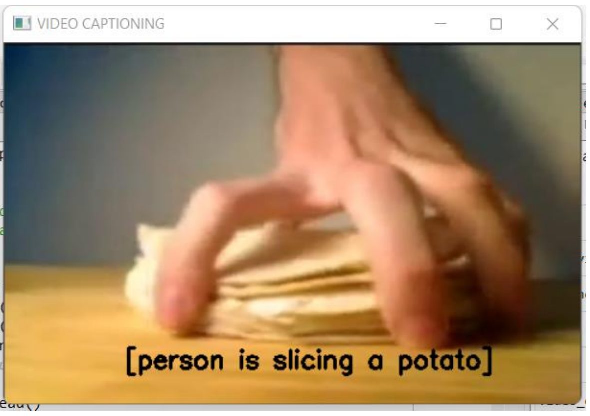
5.1.Outcomes and Screenshots



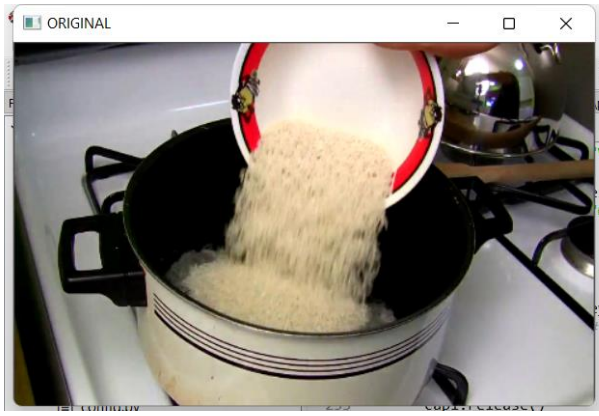


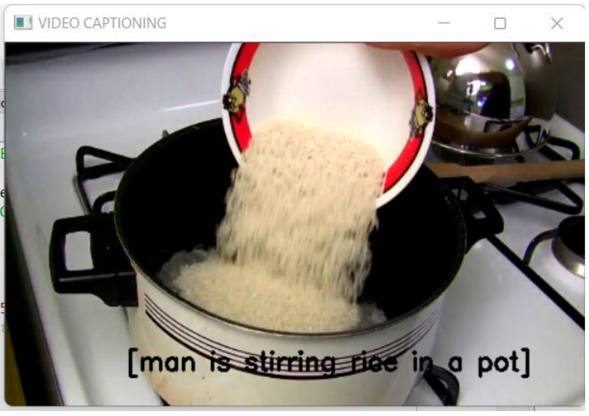
Original video Frame video caption detected frame





Original video Frame video caption detected frame





Original video Frame video caption detected frame

A. Quantitative analysis

Qualitative analysis is the statistical and mathematical approach to the research. The training and validation accuracy and loss of the proposed system during the training process is as shown in Fig.5.3. and Fig.5.4. respectively. The convolutional neural network with VGG 16 transfer learning model has been used to train the network for video. For the training process, 150 epochs were set. From the graph shown in Fig. 5.3, it is observed that the training accuracy goes on increasing with the increase in the epoch. In the same manner, validation accuracy also increases parallel. From the loss graph, as shown in Fig 5.4., It is also observed that as the epoch goes on increasing loss is decreasing.

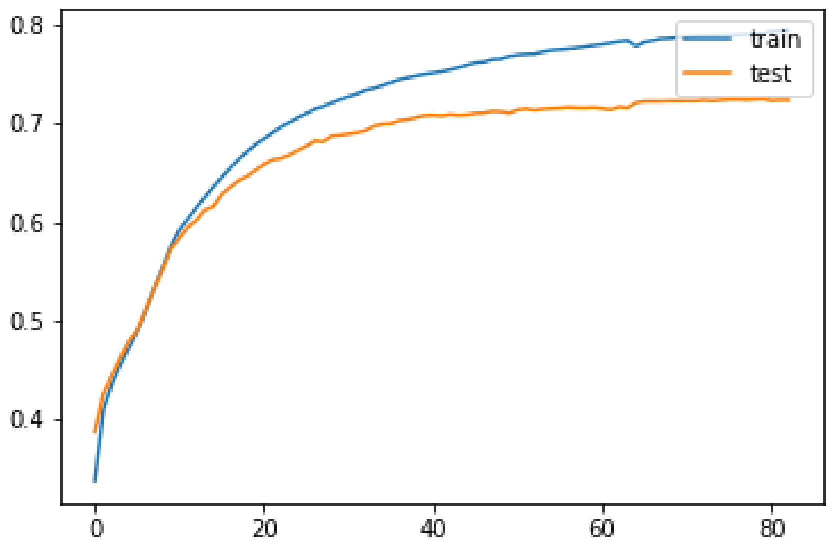


Fig. 5.3 Accuracy progress during the training process

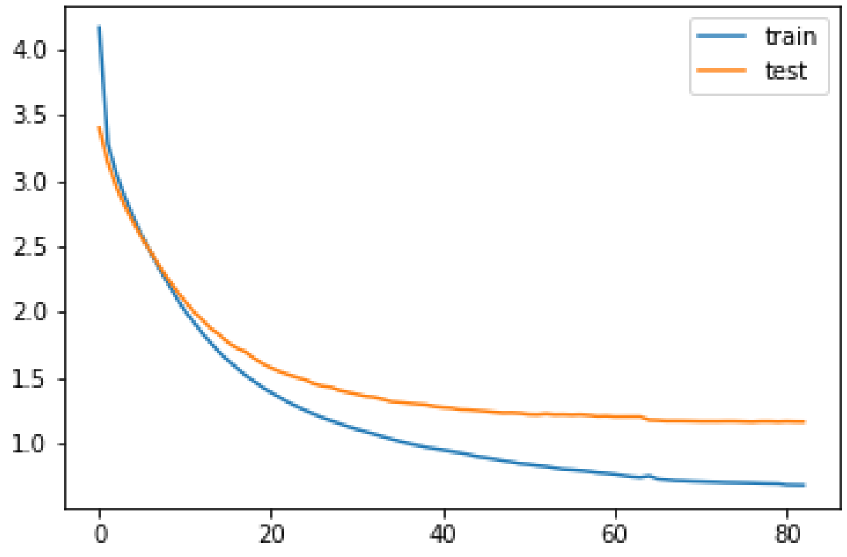


Fig. 5.4 Loss progress during the training process

CHAPTER 6
CONCLUSION

This chapter concludes the report on the Video Captioning using Keras System by stating the final overview and usefulness of the project in society. It talks about what is the aim of the project. Next, it also gives a brief idea about the applications of the project, limitations of the project, and also what kind of future work can be carried out or easily integrated into this project. These features work to enhance the total work of the project.

6.1. Conclusion

In recent years, many models have been proposed and presented to generate captions for images and short videos. Although these models are helping to advance the technology, they suffer from inaccuracies due to fundamental constraints, resulting in limited use in practical situations.
This paper proposed a novel approach to the video description. In contrast to related work, we construct descriptions using a VGG16 model. where frames are first to read sequentially and then words are generated sequentially by using the LSTM network. This authorizes us to work on variable-length input and output the same modelling temporal structure. Using our model significantly benefits from additional data, suggesting that it has a high-capacity model.
The system is evaluated with the MSVD dataset and we get an accuracy of 79.40 %.
The Video captioning problem is not solved yet if we consider the performance is so far from human-level captioning. we identify several possible future directions, according to suggestions in the literature and progress in related fields

6.2. Future Work

1. Adding attention blocks and pretrained embedding like gloves so that the model understands sentences better.
2. Right now, the Model uses 80 frames here improvement needs to be done so it works on longer videos.
3. Adding User Interface to the project.

6.3. Limitations

1. The system can be useful only for short clips.

6.4. Applications

1. Video captioning used to know the context of video
2. Easy to understand video.

3. YouTube and other media platforms.
4. Instead of using the features given extracting more features on my own using models like I3D specifically designed for videos.
5. Captions make it easier to catch and keep your audience's attention, as demonstrated by the fact that videos with captions have a longer viewing time.

REFERENCES

[1] Y. Yu, H. Ko, J. Choi, and G. Kim, "End-to-end concept word detection for video captioning, retrieval, and question answering," in Proceedings of IEEE Conference on Computer Vision.

[2] Subhashini Venugopalan UT Austin vsub@cs.utexas.edu; Lisa Anne Hendricks UC Berkeley lisaanne@berkeley.edu; Raymond Mooney UT Austin mooney@cs.utexas.edu; Kate Saenko Boston University saenko@bu.edu.

[3] Yingwei Pan, Ting Yao, Houqiang Li, and Tao Mei University of Science and Technology of China, Hefei, China Microsoft Research, Beijing, China panyw.ustc@gmail.com, {tiyao, tmei}@microsoft.com, lihq@ustc.edu.cn

[4] Anna Rohrbach; Marcus Rohrbach; Niket Tandon; Bernt Schiele;

[5] Max Planck Institute for Informatics, Saarbr. ucken, Germany UC Berkeley EECS and ICSI, Berkeley, CA, United States

[6] Rakshith Shetty and Jorma Laaksonen Department of Computer Science Aalto University School of Science P.O.BOX 15400, FI-00076 AALTO Espoo, Finland.

[7] Kyunghyun Cho Bart van Merrienboer Caglar Gulcehre; Dzmitry Bahdanau Jacobs University, Germany;Fethi Bougares Holger Schwenk ;Yoshua Bengio Universite du Maine, France

[8] Pingbo Pan; Zhongwen Xu; Yi Yang Fei Wu; Yueting Zhuang; Zhejiang University of Technology Sydney. {lighnt001,zhongwen.s.xu}@gmail.com yi.yang@uts.edu.au {wufei,yzhuang}@cs.zju.edu.cn

[9] Bairui Wang;Lin Ma; Wei Zhang; Wei Liu; Tencent AI Lab School of Control Science and Engineering, ShandongUniversity

[10] Word Beam Search: A CTC DecodingAlgorithm https://towardsdatascience.com/word-beam-search-a-ctc-decoding-algorithm-b051d28f3d2e

[11] Islam, Saiful & Dash, Aurpan & Seum, Ashek & Raj, Amir & Hossain, Tonmoy & Shah, Faisal. (2021). Exploring Video Captioning Techniques: A Comprehensive Survey on Deep Learning Methods. SN Computer Science. 2. 10.1007/s42979-021-00487-x.