iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** V **Month of publication:** May 2022

DOI: https://doi.org/10.22214/ijraset.2022.42506

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC VALUE: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue V May 2022- Available at www.ijraset.com

Video Captioning Using Neural Networks

Prathamesh Padmawar[1], Ram Borade[2], Aditya Hol[3]
[1, 2, 3]*Final Year Undergraduate Students, Department of Electronics and Telecommunication Engineering, Pune Institute of Computer Technology*

Abstract: ***Researchers in the fields of computer vision and natural language processing have been concentrating their efforts in recent years on automatically developing natural language descriptions for videos. Although video comprehension has a variety of applications, such as video retrieval and indexing, video captioning is a difficult topic to master due to the complex and diverse nature of video content. Understanding the relationship between video content and natural language sentences, on the other hand, is still a work in progress, and several approaches for improved video analysis are being developed. Because of their superior performance and high-speed computing capabilities, deep learning approaches have shifted their focus to video processing. This research aims at the end-to-end structure of a deep learning based encoder-decoder network for creating natural language descriptions for video sequences. The use of a CNN-RNN model paired with beam search to generate captions for the MSVD dataset is explored in this study. We have compared the results with beam search and greedy search approach.***
The generated captions from this model is generally grammatically incorrect. Our paper focuses on improving those grammatical errors using encoder-decoder model. Grammatical errors include spelling mistakes, incorrect use of articles, prepositions, pronouns, nouns, etc or even poor sentence construction. Using beam search for k=3, the captions generated by our algorithm get a BLEU score of 0.72. After passing the generated captions through a grammar error correction mechanism, the results improve to a BLEU score of 0.76. The results increased by 5.55% after grammar correction. The blue score reduces as the value of k decreases, but the time it takes to generate captions decreases as well.
Index Terms: Video captioning, end-to-end structure, MSVD dataset, encoder-decoder model, beam search, grammar correction

I. INTRODUCTION

Text, music, image, video, and other forms of multimedia are all part of today's digital material. With the spread of sensor-rich mobile devices, video has become a new means of communication between Internet users. Video data has been generated, published, and distributed rapidly as a result of the significant rise in Internet speed and storage capacity, and it has become an integral aspect of today's big data. Improved methodologies for a wide range of video comprehension applications, including online advertising, video retrieval, and video surveillance, have resulted as a result of this. Understanding video contents is a critical challenge that supports the success of these technological advancements. In this paper, we are focusing on video captioning using deep neural networks. The goal of video captioning is to make a complete and natural sentence out of a single label in video classification, capturing the most informative dynamics in videos. Template based models takes structured data as input and generates the natural language sentences that describes the data in human manner. The main drawback of these models is that it predefines the rigid sentence structure. For video captioning, there are two basic approaches: template-based language models and sequence learning models (e.g., RNNs). The first divides the sentence into different sections and predefines the unique rule for language grammar (e.g., subject, verb, object). Many works match each portion with recognized words from visual content by object recognition and then create a sentence with linguistic limitations using such sentence fragments. Sequence learning methods are used by the latter to learn a direct mapping between video content and texts. Hindi-language captions are also being generated by converting the English captions. In our project we have used sequence learning end-to-end encoder-decoder model.
This model has 2 parts:-

1) *Encoder:* The video features are encoded into an array using this method. In our project, we capture 80 video frames and extract 4096 features for each frame 1 using the VGG16[10] CNN model, which has been pre-trained. As a result, we have an array with dimensions of 80 X 4096.
2) *Decoder:* The hidden features generated by the encoder are fed into the decoder, along with the input captions, to generate video captions. Because this is a sequential task, an LSTM[?] chain is used.

In our project, CNN-RNN[9] model is used with beam search to create video captions. The data is first pre-processed, and then the features from the videos are extracted using a pre-trained VGG16 CNN model. Each video is cut into 80 frames, which are then run through the VGG16 model. The training model is then created using LSTM.

II. THEORETICAL BACKGROUND

The goal of video captioning is to automatically generate natural-language descriptions of videos, which is a joint task of computer vision and natural-language processing. Video captioning plays a crucial role in many real-world applications, such as fast content-based video retrieval, video understanding, assist device for the visually impaired and automatic subtitle generation system. The traditional encoder-decoder framework, e.g., sequence to sequence: video-to-text (S2VT), has achieved promising performance on many sequences generation tasks, including machine translation, dialogue system, image and video question answering and even image and video captioning. In such a framework, visual information is commonly encoded by the convolutional neural network (CNN) or recurrent neural network (RNN). Then, RNN is used to decode caption sentences. However, the caption sentences generated by the one-step encoder decoder model ways involve many word errors and grammar errors.

III. RELATED WORK

Yang et al. [1] Adversarial LSTM for Video Captioning. In this project, GAN (Generative Adversarial Network) is used. GAN is distinguished by the interactions between two operations, the generator and the discriminator. As a result, the LSTMGAN paradigm outperforms Classifier models by a large margin. The paper's main flaw is that it does not employ reinforcement learning. The use of reinforcement learning can help boost productivity even more.

Zhao et al. [2] Built on a Co-Attention Model, Recurrent neural network for Video Annotations (CAM-RNN). A rnn based on co-attention model is constructed in this paper. Co- Attention Model is used to encapsulate the textual and visual elements. RNN operates as a decoder to create the video description. CAM is made up of a visual recognition module, a textual attention component, and a regulating gates (Co-Attention Model). During the production process, the visual recognition unit may adapt to the important elements of each frame as well as the images which are most connected to the captions. The textual attentiveness component can dynamically concentrate on most significant phrases and words that have been earlier created. Likewise, a balancing gate is established between the two attention modules to regulate the relevance of textual and visual aspects while constructing the description. The paper's main flaw is that when the dataset grows increasingly complicated, efficiency suffers. Taking the MPII-MD dataset as an illustration. It comprises 68000 video segments from 94 different movies.

Wang et al. [3] Deep learning algorithms are used to merge picture and audio in video captioning. To extract characteristics from image, audio, and semantic information, the suggested system employs a variety of neural networks. Audio and image characteristics are combined before being sent into a long short-term memory (LSTM) for activation. The integrated audio-image aspects help construct a more performant network by assisting the overall semantics. The paradigm for video description provided in this paper might be researched further to boost performance.

Pretraining more unique audio streams can enhance the effectiveness of audio feature extraction much farther. Two different CNN models are utilised to learn audio events and occurrences.

Oura et al. [4] A multimodal deep neural network containing picture sequence characteristics was employed for video captioning. MDNNiSF is a method for creating a caption of a short video that is discussed in the paper. Integrating S2VT with NeuralTalk2, an image captioning algorithm due to its capacity to understand connections between text and picture segments and hence provide correct descriptions. MDNNiSF beats S2VT in testing employing two video caption data sets, MSVD and MSR-VTT. The primary issue in this study is that it fails to account for alignments acquired by NeuralTalk2 between text segments and picture fragments. Hierarchical Recurrent Neural Networks surpass MDNNiSF in METEOR. MDNNiSF is orthogonal to this approach, thus it must be extended in the same direction.

Nguyen et al. [5] A multistream hierarchical boundary network is used for video captioning. This paper introduces the Multistream Hierarchical Boundary (MHB) model, which combines a fixed hierarchy recurrent architecture with a soft hierarchy layer to define clips using intrinsic feature boundary cuts within a video. Border encoding is used to represent a video. Parametric Guassian attention is paid to films of varying lengths.

The intrinsic properties of videos are leveraged to construct a flexible hierarchical video description. This model has been trained from beginning to end for video description.

The approach uses intrinsic feature boundary cuts inside a movie to define segments. The method's accuracy will be harmed, however, because boundary cuts are not always obvious in high-resolution recordings.

Shin et al. [6] The investigation in this study uses videos with more than one sentence. The project presents to build video subtitles that communicate richer information by temporally segmenting the video with action localisation, synthesising several captions from diverse frames, and combining them with natural language processing methods.

Caption Generation is divided in three steps :

1) Temporal Segmentation with Action Localization
2) Backward Coreference Resolution
3) Connective Word Generation.

The problem with the work is that using an existing dataset to train an action classifier limits the number of action classes. It could be a better idea to unsupervised localise the actions. It's hard to tell whether 'a man' from two different frames refers to the same person or not using the existing pipeline. As a result, the present system will have to be combined with the facial recognition system.
Lee et al. [7] Using Video Captioning to Capture Long-Range Dependencies. In this paper, the temporal capacity of a video captioning network with a non-local block is examined.
It provides a non-local block video captioning method for capturing longrange temporal dependencies. Local and non-local features are used separately in the suggested model. It makes use of both types of characteristics. It tests the method using a dataset from the Microsoft Video Description Corpus (MSVD,YouTube2Text). The results suggest that in video captioning datasets, a non-local block applied along the temporal axis helps alleviate the LSTM's long-range dependency problem. The primary flaw in this paper is that the model's sentences are not optimised. During training, reinforcement learning can be utilised to directly optimise the phrases generated by the model.
Xiao et al. [8] With a Temporal And Region Graph Convolution Network, video captioning is achieved. With Temporal Graph Network (TGN) and Region Graph Network (RGN), this research proposes a revolutionary video captioning paradigm (RGN). The temporal sequential information and the region object information of frames are encoded using a graph convolution network. TGN primarily focuses on leveraging the sequential information of frames, which is often ignored by most existing approaches. RGN was created to investigate the connections between important objects. The prominent boxes of each frame are extracted using an object detection algorithm, and then a region graph is built based on their placements. The context features that are given into the decoder are acquired using an attention method.
Hoxha et al. [9] In this paper, a new framework for picture captioning is proposed that combines creation and retrieval. For remote sensing image captioning, a new CNN RNN framework has been developed. Multiple captions for a target image are generated using a CNN-RNN architecture paired with beam search. The best caption is then chosen based on linguistic similarity to reference captions for the most similar photographs. The RSCID dataset is used to generate the results.

IV. TECHNICAL APPROACH

For every machine learning related problem we need a good dataset to get proper and expected results. So firstly, the dataset is chosen. There were two main choices1) MSVD dataset and 2) MSR-VTT. We chose the first MSVD as that was small and had 1450 video clips with 41 sentences for each clip.
As we all know videos are made up of multiple images arranged one after another called frames. At first 80 images are extracted from each input video. 4096 features are extracted from a single frame. Pretrained CNN model VGG-16 is used for extracting an 80x4096 numpy array for each video.
Next step is to choose a training model to train data. Our input data is in sequence of features of frames. To process sequences of frames for each video usually sequence models are used. sequence models are machine learning models that input or output sequences of data. Among LSTM+GAN and LSTM+CNN, later one LSTM - an artificial recurrent neural network (RNN) is used for training. This process runs in a block called encoder. To make the model more accurate and efficient, a reinforcement polishing network can be used. These models are usually trained on millions of samples making them good at object detection. Later all we need to do is adjust a few layers and train on our custom data.
In the decoding phase, input of one LSTM cell is given to the next cell. Thus, this results in combining captions to give complete sentences.

V. TECHNICAL SPECIFICATION

The video captioning task aims to describe video content using several natural-language sentences. Although one-step encoder-decoder models have achieved promising progress, the generations always involve many errors, which are mainly caused by the large semantic gap between the visual domain and the language domain and by the difficulty in long-sequence generation. The underlying challenge of video captioning, i.e., sequence-to-sequence mapping across different domains, is still not well handled. This project will aim toward generating captions and provide the accuracy of the captions.

ISSN: 2321-9653; IC VALUE: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue V May 2022- Available at www.ijraset.com

VI. SYSTEM DESIGN

A. Block Diagram

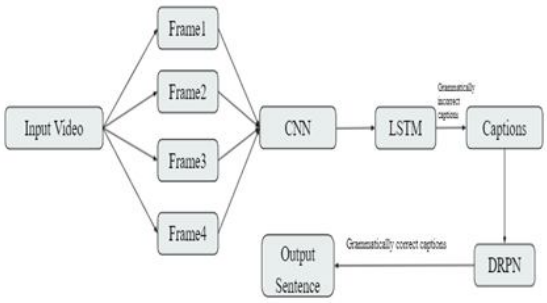


Fig . 6.1.1 Block Diagram

B. Data Collection

The MSVD[2] dataset set by Microsoft is being used for this study. This data set includes 1970 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. There are two directories in the dataset: training_data and testing_data. 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 is also a training_label and testing_label json files. The captions for each ID are contained in these json files.

The MSVD corpus was created from a compilation of Youtube video clips that every record a single activity by one or even more people. Then every video segment is represented by a single sentence. We began by deleting any explanations that were not in English or had misspelt phrases. Then we converted all of the faulty descriptors to lower case tokens. For unedited video, we take images every 10 frames.

Each video was divided into multiple sections in MSVD, each one corresponded to one or even more captions. We separated the data set into two portions by respective video names, producing Training and Testing sets with an 8:2 ratio, to avoid parts from the very same video (despite when they may capture different behaviours) showing in both training and testing sets, and therefore biassing up our assessment scores.

C. Feature Extraction

Each video in the dataset is treated as a series of frames of images to extract the video's features. All images from the video stream were recovered piece by piece. 1231 Depending on the duration of the video, the frame rate captured differs. As a consequence, again for purpose of simplicity, only eighty frames from every movie are chosen. Each one of the eighty frames is analyzed by a VGG16 that has been trained before, which yielded 4096 attributes. The VGG16 convolutional neural network design emphasises 3 × 3 filtered convolutional layer with step 1, while utilises the very similar padding and maxpool structure as a 2x2 filtered having step 2. Both convolutional and maximum pooling tiers are placed in the very same manner all through the layout. The result is finished by 2 fully connected layers and a softmax. The (80, 4096) shaped array is formed by stacking these characteristics. The overall number of frames generated from the video is 80, with each frame yielding 4096 unique features.

D. Preprocessing of Captions

The captions and video IDs are stored in two columns in the train_list. The train list's beginning captions are labelled 'bos,' while the end captions are labelled 'eos.' Following that, the data is divided into training set and validation sets as follows:- training list contains 85% of the data, whereas the validation list contains the 15% of the data.

Because we simply tokenize the words in the training data, the vocab list only contains the captions from the training_list. The padding of the sentences is done after tokenizing to make all the sentences of the same word length. We have inflated all of them to be ten words in our project.

The caption with the most words in the entire data set has 39 words, but the majority of captions have between 6 and 10 words. The captions are filter out to make the dataset outlier free because then we need to pad the sentences with whitespaces to the length of 39 words. This can result in wrong prediction. If most sentences are 10 words long and we have to pad them to double their length, there will be a lot of padding. These heavily padded sentences will be utilised for training, resulting in the model predicting padded tokens almost exclusively. Because padding essentially entails the addition of white spaces, most of the sentences predicted by the model will simply have more blank spaces and fewer words, resulting in incomplete phrases.

For the captions, we only used the top 1500 words as our vocabulary. Any captions that are generated must be included in the 1500-word limit. Despite the fact that there are over 1500 distinct terms, most of them appear just once, twice, or three times, making the vocabulary susceptible to outliers. As a result, we only chose the top 1500 most frequently occuring words to keep it secure.

E. Long Short-Term Memory Networks

Recurrence equations are used by mapping input sequences to hidden states and subsequently creating outputs, traditional RNNs are utilised to learn intricate temporal dynamics.

$$h_t = \gamma(W_h x_t + U_h h_{t-1} + b_h) \quad (3.1)$$

$$o_t = \gamma(U_h h_t + b_o) \quad (3.2)$$

Equation 3.1 and 3.2 is an element-wise non-linear function, such as RELU or hyperbolic tangent, where W_h, U_h denote the weight matrices and b denotes the bias, and is an element-wise non-linear function. ot is the output at time t, ht is the hidden state with M hidden units, and x_t is the input.

Text generation and speech recognition have been demonstrated to be successful using traditional RNNs. Longrange temporal associations movies, on the other hand, are difficult to manage due to the gradient problem of inflating and fading. Hochreiter and Schmidhuber[17] devised the LSTM network, which has been shown to successfully prevent gradient vanishing and explosion problems during backpropagation through time (BPTT).

This one is due to the network's inclusion of memory blocks, which allow it to acquire lengthy temporal relationships, forgetting formerly concealed state, and modify them with fresh content. LSTM, as implemented in our framework, consists of many controlling gate and a storage cell. At every period interval, have xt, ct, and ht denote the inputs, cell memories, and hidden control values, accordingly. The LSTM will calculate the hidden control sequence (h1,..., hT) and the cell storage series given a set of input (x1,..., xT) (c1, ..., cT).

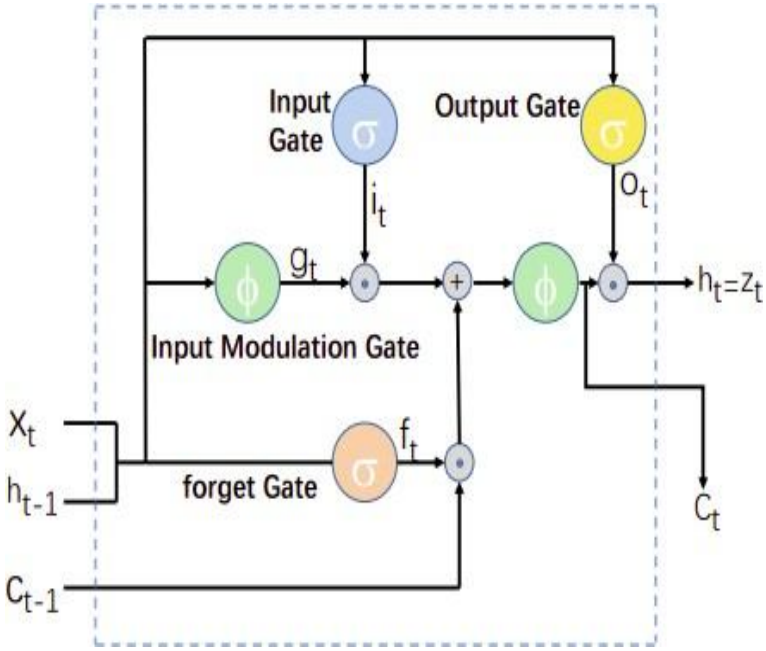


Fig : 6.5.1 Basic LSTM memory cell.

F. LSTMs for Sequence Generation

A recurrent neural network (RNN) is a generalized feed forward neural network that has sequences added to it. Based on the following recurrences, standard RNNs learn to convert a sequence of inputs x1, ..., xt into a succession of hidden states h1, ..., ht , and from the hidden states to a sequence of outputs z1, ..., zt :

$$h_t = f(W_{xh}x_t + W_{hh}h_{t-1}) \quad (3.3)$$

$$z_t = g(W_{zh}h_t) \quad (3.4)$$

In the equations 3.3 and 3.4 , input taken is a fixed length vector which is represented by x_t . *f* and *g* are non-linear functions. By non-linear functions, we mean it can be hyperbolic or sigmoid function or any other non-linear function. Hidden state is represented by h_t . Weights connecting neurons and layers are represented by W_{ij} and output vector is represented by z_t .RNNs can train to map sequences when the orientation between the outputs and inputs is given beforehand, but it's uncertain if they can solve issues with fluctuating size inputs (x_i) and outputs (z_i). This issue can be fixed by training one RNN to translate input sequence data to a fixed size vector, followed by another RNN mapping the vector to an output vector. Another well-known issue with RNNs is the difficulty of teaching them to acquire long-term dependencies. LSTMs with intentionally programmable memory blocks, on the other hand, have been shown to be capable of learning long-range temporal relationships.

A storage cell c is at the heart of the LSTM model, encoding the learning of the data gathered up until that point at each time step. The cell is controlled by sigmoid function gates with a range of [0,1] that are performed multiplicatively. These gates determine if the LSTM retains (if a layer evaluate to 1) or throws away the result from the gate (if it evaluates to 0). The three gates allow the LSTM to learn complicated long-term correlations: output gate (o) determines how much of the memory to transfer to the hidden state ht , forget gate (f) allows the LSTM to forget its previous memory c_{t-1}, and input gate I determines if the LSTM considers its current input x_t .

G. Building the Model

An encoder-decoder architecture is the chosen model for most text generating challenges. We'll employ this sequence-to-sequence architecture in our problem statement since text must be generated. One thing to keep in mind in this architecture is that the encoder cell's final state always serves as the decoder cell's initial state. The encoder was utilized to input the video features in our scenario, and the decoder was supplied the captions. We have used sequence to sequence model in which LSTM is used for encoder.

The encoder, intermediate (encoder) vector, and decoder are the three components of our model Encoder :

1) A stack of many recurrent units (for higher performance, LSTM or GRU cells) that each accepts a single element of the input sequence, accumulates information for that element, and propagates it forward.

2) In our model, the input sequence is collection of all the frames of the videos. Each frame is represented as x_i where i is the order of that word.

3) The hidden states h_i are computed using the formula:

$$h_t = f(W^{(hh)}h_{t-1} + W^{(hx)}x_t) \quad (3.5)$$

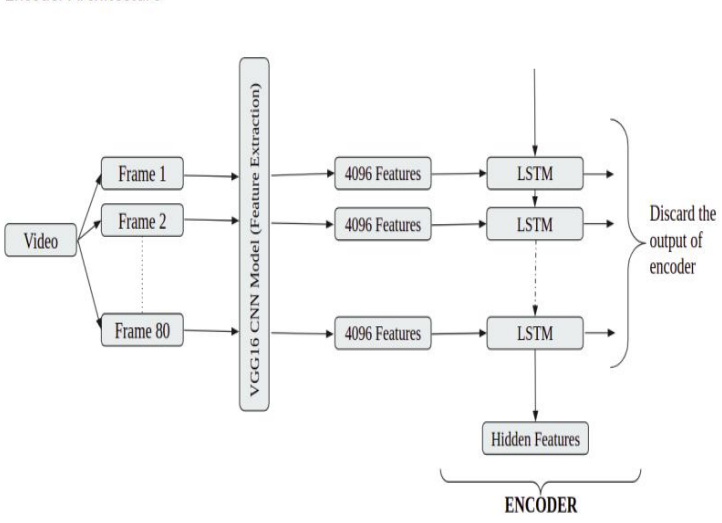


Fig 6.7.1: Encoder Architecture

a) Encoder Vector

- This is the model's final hidden state, generated by the encoder. The equation 3.5 is used to calculate it.
- This vector seeks to incorporate all input element information in order to aid the decoder in making correct predictions.
- It serves as the model's initial hidden state for the decoder.

b) Decoder

- A stack of recurrent units, each of which predicts an output y_t at a given time step t.
- Each recurrent unit takes in a hidden state from the preceding unit and outputs as well as its own hidden state.
- The output sequence are the captions generated for the frames.
- The formula is used to calculate the output y_t at time step t:

$$y_t = \text{softmax}(W^s h_t) \qquad (3.6)$$

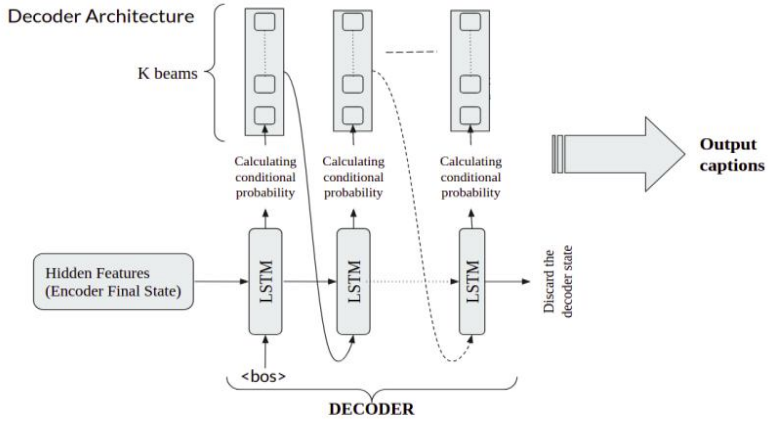


Fig 6.7.2 : Decoder Architecture

We compute the outputs by combining the hidden state at the current time step with the weight W(S). Softmax is used to generate a probability vector that will aid in the prediction of the ultimate result.
Initially, first frame's features are supplied into the encoder's first LSTM cell. This is followed by the second frame's features, and so on until the 80th frame. Because we're only interested in the encoder's end state, all of the encoder's other outputs are ignored. Final state of the encoder is used as the decoder's initial state. The first decoder uses LSTM as input to begin the sentence. Each word of the caption from the training data is fed one at a time until is reached.
The encoder's time steps are equal to the number of LSTM cells we chose for the encoder, which is 80. The amount of features from video encoder tokens is 4096 in our situation. The decoder's time steps are equal to the number of LSTM cells in the decoder, which is ten, and the number of tokens is equal to the vocabulary length, which is 1500.
The final model looks as follows :

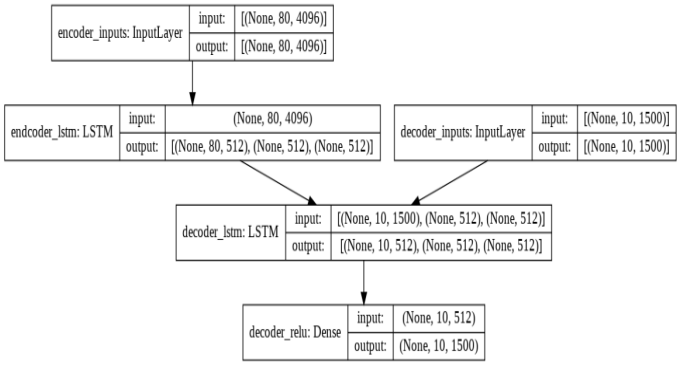


Fig 6.7.3: Model Architecture

We tested our model by running it on different number of epochs i.e, 25, 50, 75, 150.

H. Generation of Captions

The vocabulary of candidate sequence is scored on the basis of their likelihood. Two approaches are used to predict the possibility of the next word that can be which is, greedy search and beam search.

1) *Greedy Search:* After analyzing and extracting the feature, we employed the use of decoder to perform the task of estimating the probability of occurrence of consecutive terms of the video caption from the vocabulary. In decoder component we implemented greedy search algorithm to effectively estimate probability of terms. The main goal of this approach is to compute the term with maximum possible conditional probability term from the given list of known terms. The algorithm terminates when the sequence of terms selected by the greedy search reaches a limit say 'L'. This satisfies the algorithm's condition and helps terminate, thereby resulting an output sequence and is assigned as the video caption for te given input encoded features and vocabulary.
2) *Beam Search Decoder:* Beam search is an approach that explores all possible choices unlike the greedy search, which focuses only on the most probable option. The amount of choices can be limited with the help of a parameter 'k', when specified the algorithm performs k number of parallel searches and then the algorithm comparatively analyses the various selections or choices. We experimented k values with '2' and '3'. We observed that as the value of 'k' increases the time complexity varies exponentially.

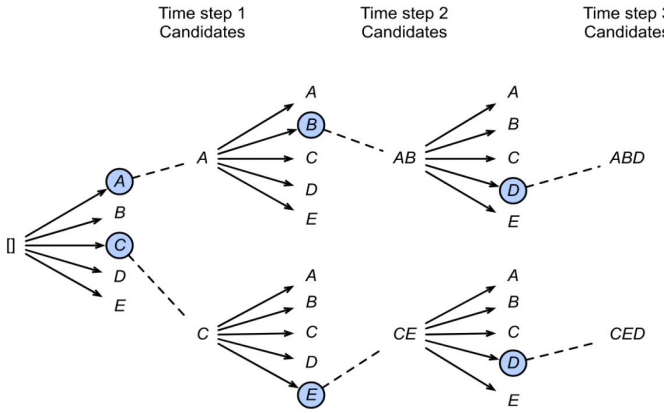


Fig 6.8.1 Beam Search Decoder

I. Grammar Error Correction

The captions generated from the encoder-decoder model are usually grammatically incorrect. So, we propose Grammar Error Correction(GEC) network to detect and correct those errors. We have used Sequence to Sequence learning model for the implementation.

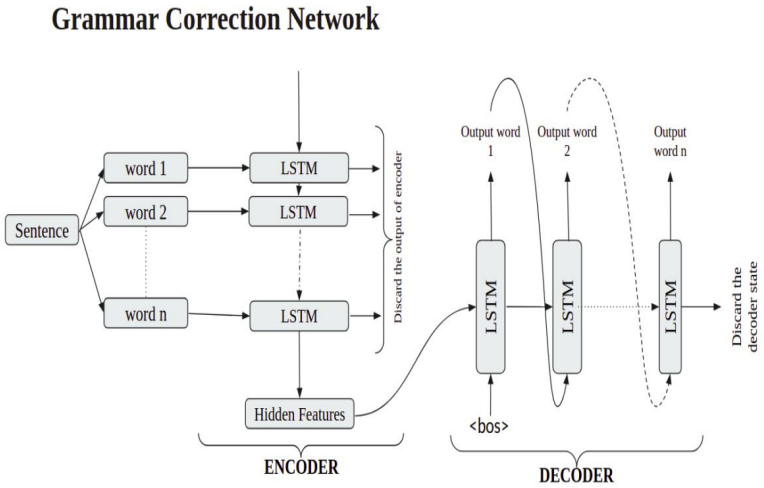


Fig 6.9.1: Our Grammar Error Correction Network

VII. RESULTS

We trained our model using LSTM and we have noted the loss and accuracy for each epochs.

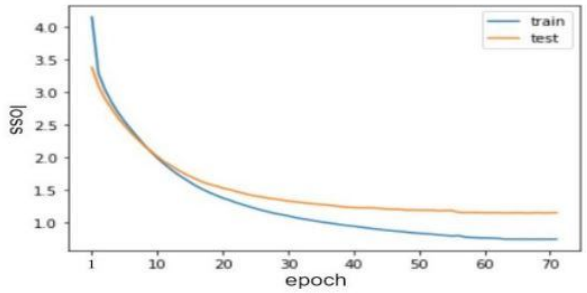


Fig 7.1 : Loss for each epoch

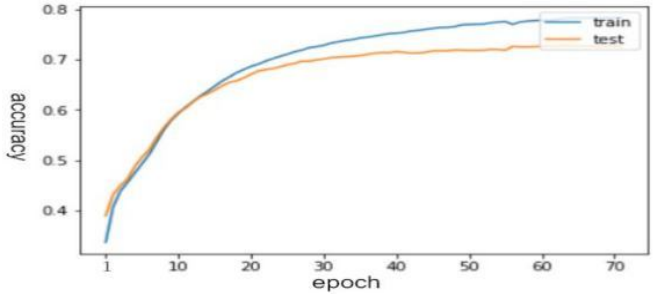


Fig 7.2 : Accuracy for each epoch

The training model loss and accuracy for the training and testing datasets are shown in Figures 7.1 and 7.2, respectively. The lines in the preceding graph are approaching a constant value, which is why the training is ended after approximately 78 epochs. To generate the best accurate captions for a video, we combined encoder decoder architecture with beam search. In comparison to greedy search, beam search produces more accurate results; however, beam search takes longer to process. We can utilize greedy search to get faster results in real-time prediction.

Table 7.1: Sample Results using beam search

Video	Result using Beam search (k=3)	Result using beam search(k=2)
	a person is cooking eggs in a pan	a person is cooking in pan
	a cat is playing a piano	a cat is playing a piano
	a dog is playing with a ball	a dog is playing
	a man is boiling water	a man is boiling

The captions created using beam search for k=2 and k=3 are shown in table 7.1. When k=2 is used instead of k=3, the captions created are less accurate. In the first example, captions generated with k=2 are less grammatically correct than captions generated with k=3. Example 3 in table 7.1 is an example of incorrect caption generating. The right title for that video would have been "Men are running," but instead "Men are playing soccer" and "Men are playing a ball" were generated. This occurs because the accuracy of our model suffers when the main object in the image is small. The size of the significant object in the first two photographs is large, and it is generating correct captions.

Table 7.2 Results with and without GEC

Video	Result before GEC with Beam Search (k=3)	Result after GEC with beam search(k=3)
	a puppy playing ball	a puppy is playing a ball
	a person cutting with knife	a person is cutting with knife

In Table 7.2 we have compared the results of the captions generated by using grammar error correction network with beam search and captions generated without grammar correction network using greedy search is less then the captions generated using beam search with k=2 and k=3.

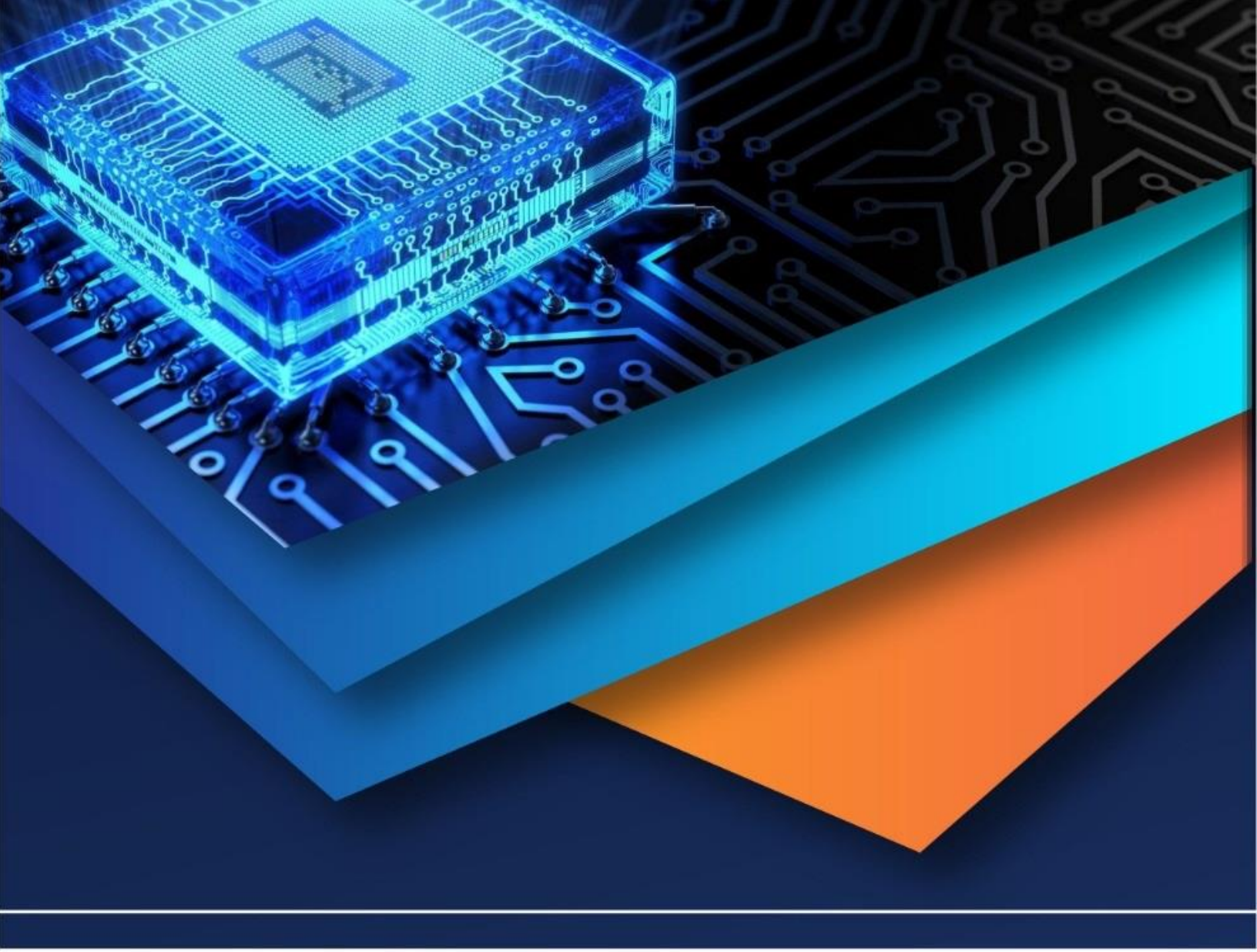
VIII. CONCLUSION

We concentrated on creating subtitles for the video dataset in this project. Initially, we concentrated on developing and training our model using the LSTM encoder. The video datasets features were extracted, the captions were analysed, and the model was built. To create the captions, we employed a variety of search algorithms and analysed their correctness. We discovered that beam search is more accurate than greedy search, and that the accuracy grows as the node value in the beam search increases. The time it takes to generate a caption increases as the number of nodes in a beam search increases. We built the Grammar Error Correction network to correct the Grammatical Errors in the generated Captions. We used encoder-decoder architecture to create the model. BLEU Score of the generated captions increases after the captions are passed through Grammar Error Correction Network. Our methodology results in a 5.55% boost in the quality of generated captions. Furthermore, we've discovered that as the number of beams in the beam search rises, the time it takes to generate captions climbs exponentially, and the more beams there are, the more accurate the model's results become.

REFERENCES

[1] Yang Yang, Jiangbo Ai, Yi Bin, Alan Hanjalic, "Video Captioning by Adversarial LSTM", IEEE Transactions on Image Processing 2019
[2] Bin Zhao, Xuelong Li ,Xiaoqiang Lu , "CAM-RNN: Co-Attention Model Based RNN for Video Captioning", IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 28, NO. 11, NOVEMBER 2019
[3] Chien-Yao Wang, Pei-Sin Liaw, Kai-Wen Liang, Jai-Ching Wang, Pao-Chi Chang, "Video Captioning Based On Jonit Image–Audio Deep Learning Techniques", 2019 IEEE 9th International Conference on Consumer Electronics (ICCE-Berlin)
[4] Soichiro Oura, Tetsu Matsukawa, Einoshin Suzuki, "Multimodal Deep Neural Network with Image Sequence Features for Video Captioning", 2018 International Joint Conference on Neural Networks (IJCNN)
[5] Thang Nguyen, Shagan Sah, Raymond Ptucha, "Multistream Hierarchical Boundary Network For Video Captioning", 2017 IEEE Western New York Image and Signal Processing Workshop (WNYISPW)
[6] Andrew Shin, Katsunori Ohnishi, Tatsuya Harada, "Beyond Caption To Narrative: Video Captioning With Multiple Sentences", 2016 IEEE International Conference on Image Processing (ICIP)
[7] Jaeyoung Lee, Yekang Lee, Sihyeon Seong, Kyungsu Kim, Sungjin Kim, Junmo Kim, "Capturing Long-range Dependencies In Video Captioning", 2019 IEEE International Conference on Image Processing (ICIP)

[8] Xinlong Xiao, Yuejie Zhang, Rui Feng, Tao Zhang, Shang Gao, Weiguo Fan, "Video Captioning With Temporal And Region Graph Convolution Network", 2020 IEEE International Conference on Multimedia and Expo (ICME)
[9] Genc Hoxha, Farid Melgani, jacopo Slaghenauffi, "A New CNN RNN Framework For Remote Sensing Image Captioning", 2020 Mediterranean and Middle-East Geoscience and Remote Sensing Symposium (M2GARSS)
[10] Jiahui Tao, Yuehan Gu, JiaZheng Sun, Yuxuan Bie, Hui Wang " Research on vgg16 convolutional neural network feature classification algorithm based on Transfer Learning", 2021 2nd China International SAR Symposium (CISS)



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY