Natural Language Video Captioning in Bengali Using Deep Learning

A Thesis

Submitted in partial fulfillment of the requirements for the Degree of

Bachelor of Science in Computer Science and Engineering

Submitted by

Saiful Islam	**160104040**
Amir Hossain Raj	**160104056**
Aurpan Dash	**160104065**
Ashek Seum	**160104067**

Supervised by

Faisal Muhammad Shah

Department of Computer Science and Engineering

Ahsanullah University of Science and Technology

Dhaka, Bangladesh

November 2020

CANDIDATES' DECLARATION

We, hereby, declare that the Thesispresented in this report is the outcome of the investigation performed by us under the supervision of Faisal Muhammad Shah, Department of Computer Science and Engineering, Ahsanullah University of Science and Technology, Dhaka, Bangladesh. The work was spread over two final year courses, CSE400: Project and Thesis I and CSE450: Project and Thesis II, in accordance with the course curriculum of the Department for the Bachelor of Science in Computer Science and Engineering program.

It is also declared that neither this Thesis nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.

Saiful Islam
160104040

Amir Hossain Raj
160104056

Aurpan Dash
160104065

Ashek Seum
160104067

CERTIFICATION

This Thesis titled, **"Natural Language Video Captioning in Bengali Using Deep Learning"**, submitted by the group as mentioned below has been accepted as satisfactory in partial fulfillment of the requirements for the degree B.Sc. in Computer Science and Engineering in November 2020.

Group Members:

Saiful Islam	**160104040**
Amir Hossain Raj	**160104056**
Aurpan Dash	**160104065**
Ashek Seum	**160104067**

Faisal Muhammad Shah
Associate Professor & Supervisor
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

Professor Dr. Kazi A. Kalpoma
Professor & Head
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our honourable thesis supervisor Mr. Faisal Muhammad Shah, Associate Professor, Department of Computer Science and Engineering, Ahsanullah University of Science and Technology, for his patient guidance, enthusiastic encouragement and constructive critiques of this thesis work. We would also like to thank Professor Dr. Kazi A Kalpoma, the honourable Head of the Department of Computer Science and Engineering of Ahsanullah University of Science and Technology, and all our respected teachers, lab assistants, and friends for their moral support and helpful discussions. Finally, we would like to thank our parents for their constant support and encouragement throughout our study.

Dhaka
November 2020

Saiful Islam

Amir Hossain Raj

Aurpan Dash

Ashek Seum

ABSTRACT

Video captioning is the most challenging Artificial Intelligence problem regarding the generation of natural language phrases explaining video frames' contents. It requires both understandings of images from Computer Vision (CV) domains and language model from Natural Language Processing (NLP) domains. The video captioning task remains a complicated one because of the difficulties of detecting objects and their activities from a video or sequence of frames to generate captions. It is then complicated for the language generation models to produce accurate captions from the videos. It becomes more challenging when captions are rendered in complicated languages like Bengali. There is no compatible model for Bengali caption generation, which directed us to work in this novel field of Video Captioning in Bengali. In this work, We have proposed an encoder-decoder based novel deep architecture which incorporates a combination of 2D-CNN as well as 3D-CNN as the encoder and Bi-LSTM as the decoder. We have trained and evaluated our model on the MSVD dataset and achieved a 30% and 20% score on BLEU and CIDEr, respectively. There is no such way to compare our work, as there is no previous work in this field. So, by far, we can claim our work a new start and the best one in this arena till now.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

The future development of artificial intelligence needs extensive factors that can help us understand the rich visual world around us and make communication easier for machines through natural language. With the advancement of computer technology in the recent era, significant steps have been taken toward this goal in the last few years. With the advancement of algorithms and data gathering, the concept of "Automatic Video Captioning" has gained much popularity among the researchers, where computer systems can describe the events or activities in the video in natural language while the video is being played. As videos have been the most effective tool for information storage, security surveillance, activity tracking, and many other actions, captions generated in natural language can be the most significant in analyzing the collected data. Generated natural language captions can have more significant social impacts if the captions are rendered in the native languages of different users, which leads to the task of video captioning in Bengali as Bengali being the 7th [1] largest spoken language and the first or second language of over 230 million [2] people across the world. As Bengali is a very complicated language and the field of video captioning in Bengali is yet unexplored, it is very challenging to propose a model or system that will generate captions in Bengali with considerable accuracy. The complete vacuum of research works Bengali video captioning, and the importance of Bengali caption generation calls for a compatible method to generate video captions in Bengali.

1.2 Motivation

Video is now the most preferred means of data collection in enormous fields like activity tracking, security surveillance, etc. If the generated captions from videos are improved to a

supreme level, this system can provide direction to visually impair peoples. The system can analyze videos in real-time directly from cameras and generate text about the surroundings, which will then be converted to audio to provide directions to visually impaired persons [3]. With captions in Bengali, people with Bengali as their only language can also get benefited. Robotics nowadays has reached such a stage where human-robot interaction is needed simultaneously. We need robots to express their vision in natural language so that humans can understand their way of processing [4]. However, searches on videos have not been conducted in such a way to simplify the interaction process. Generating text from videos can be an outstanding contribution to the field of human-robot interaction. Even this system can be perfected at such a level where machines can generate instruction set for humans. Then again, captioning a video in Bengali can make learning easier for people using Bengali as their mother tongue. Another application of the system can be converting sign language into natural language or text, which will help people with a hearing problem communicate with normal people [3]. In media (movies, theaters), videos are made or captured according to the script or text document. Text documents can be an excellent alternative to video data in keeping records of the activity only. This procedure can also be used for surveillance cameras as a large amount of storage is needed to store the videos. This process can save a lot of storage space by keeping the videos' information as text if captions are perfected supremely. Again, it is impossible for humans to always sit in front of monitors to check security footage for vulnerable or suspicious activities in secured areas. If video captioning is perfected satisfactorily, a real-time warning system can be prepared based on the video footage from the CCTVs as computers often surpass humans in object classification fields. It can be a significant contribution to the nationwide security system.

1.3 Objective

The main objective of this thesis work is to build a model that can extract visual features from consecutive video frames and generate natural language captions in Bengali based on the visual features, which includes objects and their spatio-temporal information. As the area of video captioning in the Bengali language is hardly touched, the first task was to select an adequate dataset depending on the videos on a variety of activities and prepare it for the task of captioning in Bengali. Acknowledging that no research work is available regarding video captioning in Bengali, we started to learn from the previous works on video captioning in English. We tried to collect some state-of-the-art methods for our Bengali caption generation model based on those works for object detection, spatio-temporal feature extraction, and language generation tasks. With this step's continuation, we tried out different combinations of object detection, spatio-temporal feature extraction, and language generation methods to face the challenge of video captioning in Bengali. Finally, depending

on the analysis based on the collected results from different combinations of methods, we proposed an efficient model gathering all the state-of-the-art methods for the tasks stated above, leading to a proper Bengali caption generation system.

Chapter 2

Literature Review

2.1 Overview

A lot of works has been done in recent years in the field of computer vision. A vast portion of these works includes different classification, description generation, and caption generation tasks from videos. But, the Bengali caption generation field is yet untouched, and no research work exists on this task. The only available works are on Bengali caption generation from images, but the number is deficient [5] [6] [7]. So, we have explored a good number of existing works of recent times on video captioning in English along with image captioning in Bengali and analyzed their working procedures to get a good idea of different parts of a whole model. To get a systematic review of the works, we have emphasized the fundamental changes made in the main architecture to improve the results of video captioning tasks. In this chapter, we have thoroughly discussed these papers and their working procedures related to video captioning n English and Image captioning in Bengali to get predictions of different methods to be used for our Bengali video captioning task.

2.2 Reviews of Related Papers

Pan et al. [8] presented a deep architecture that incorporates transferred semantic attributes learnt from images and videos into the CNN-RNN framework. Images and videos carry complementary semantics and thus can reinforce each other for captioning.

Working Approach:

- The video representation is produced by mean pooling over the visual features of sampled frames extracted by a 2-D/3-D CNN.

- Attributes from images are learnt by adopting the weakly-supervised approach of Multiple Instance Learning (MIL).
- A video MIL model is particularly devised to learn attributes from videos.
- The video representation is injected into LSTM only at the initial time.
- Attributes representations from images and videos are fed as the additional inputs into the second-layer LSTM unit.
- A transfer unit is devised to dynamically fuse them into LSTM.

Wang et al. [9] proposed Hierarchical reinforcement learning (HRL) framework for video captioning, where a high-level Manager module learns to design sub-goals. and a low-level Worker module recognizes the primitive actions to fulfill the sub-goal.

Working Approach:

- HRL framework follows the general encoder-decoder framework.
- In the encoding stage, video frame features are first extracted by a pretrained convolutional neural network (CNN).
- Frame features are passed through a low-level Bi-LSTM encoder and a high-level LSTM encoder successively to obtain low-level encoder output and high-level encoder output.
- In the decoding stage, HRL agent plays the role of a decoder, and outputs a language description.
- The HRL agent is composed of three components: a low-level worker, a high-level manager, and an internal critic.
- Manager operates at a lower temporal resolution and emits a goal, worker generates a word for each time step by following the goal, internal critic determines if the worker has accomplished the goal.

An encoder-decoder reconstruction architecture is proposed by Wang et al. [10] where CNN is used as encoder and LSTM+GRU is used for decoder part. This paper shows a possibility of backward flow (sentence to video). LSTM and GRU generates sentence fragmentation one by one and assemble them to generate sentence.

Working Approach:

- CNN architecture Inception-V4 is used as the encoder for representation of the video sequence.
- LSTM with the capabilities of modeling long-term temporal dependencies are used to decode video representation to video captions word by word.
- To exploit the global temporal information of videos, a temporal attention mechanism is employed for the decoder to select the key frames/elements for captioning.
- Backward Flow is done through Neural Machine Translation (NMT) mechanism and image segmentation.
- NMT reconstructs the source from the target when the target is achieved.

Dense video captioning involves localizing distinct events in a long video stream, and generating captions for the localized events. Xu et al. [11] proposed a model- Joint Event Detection and Description Network (JEDDi-Net) which encodes input video stream and proposes variable-length temporal events based on pooled features. A two-level hierarchical captioning module keeps track of context of temporal relationships between visual events and their captions in a single video.

Working Approach:

- 3D convolutional network (C3D) architecture is employed to encode the input frames in a fully-convolutional manner.
- Segment Proposal Network (SPN) predicts the activity proposals' start and end times.
- To compute a visual representation of each proposed event for the captioning module, predicted proposals are encoded into feature vectors.
- To model context between the generated caption sentences, a hierarchical LSTM structure is adopted.
- The high-level Controller LSTM encodes the visual context and sentence decoding history.
- The low-level Captioning LSTM decodes every proposal into a caption word by word.

Junchao Zhang and Yuxin Peng [12] proposed Attention Guided Hierarchical Alignment (AGHA) approach which exploits multi-level vision-language alignment information and multi-granularity visual features to boost the accurate generation of video captions. Vision-language alignments includes object-word, relation-phrase and region-sentence alignments.

Working Approach:

- First, multi-granularity visual features including global features, as well as region specific, relation-specific, and object-specific features are extracted.
- The features are extracted using convolutional neural network architecture - GoogLeNet with Batch Normalization that is pre-trained on ImageNet dataset.
- These features are then fed into three parallel encoder-decoder streams.
- All three streams have the same structure that includes an attention-based encoder and an alignment-embedded decoder.
- Finally, the hierarchical alignments from three streams are integrated to obtain the description sentence.

Ding et al. [13] have proposed techniques are for the application of long video segmentation, which can effectively shorten the retrieval time.

Working Approach:

- Primary task is to detect and remove the redundant frames by using STIPs in the stage of processing video frames.
- After redundant video frames have been screened, the long video is segmented by using non-linear combination of different visual elements.
- During key frame selection, the region of interest is constructed by using the STIPs that is obtained in the previous part directly.
- Finally, LSTM variant model that is combined with attention mechanism used for caption generation.

Yan et al. [14] have proposed Spatial-Temporal Attention Mechanism (STAT), that takes into account both the spatial and temporal structures in a video, so it makes the decoder to automatically select the significant regions in the most relevant temporal segments for word prediction.

Working Approach:

- Overall framework is based on the popular convolutional neural network (ConvNet) + LSTM architecture.
- 2-D/3-D Convolutional Neural Network (CNN) and Region-based Convolutional Neural Networks (RCNNs) are used to encode the video inputs to a set of fixed length vector representation.

- CNN such as GoogleNet can represent an image as a single feature vector. 3-D CNN such as C3D can represent consecutive frames as a single feature vector. R-CNN such as Faster R-CNN can represent a region or object as a single feature vector.
- Three kinds of features are fused via two-stage attention mechanism.
- Spatial attention mechanism firstly makes the decoder to select local features with more spatial attention weights, which represent the significant regions.
- Temporal attention mechanism makes the decoder to select global and motion features.

Park et al. [15] have proposed a multi-discriminator "hybrid" design, where each discriminator targets one aspect of a description.

Working Approach:

- Adversarial Inference for video description progressively sample sentence candidates for each clip, and select the best ones based on a discriminator's score.
- A "hybrid discriminator" is proposed which combines three specialized discriminators: one measures the language characteristics of a sentence, the second assesses its relevance to a video segment, and the third measures its coherence with the previous sentence.
- The proposed approach to multiple baselines is compared on a number of metrics, including automatic sentence scores, diversity and repetition scores, person correctness scores.

Rahman et al. [16] have aimed to outline an automatic image captioning system in Bangla, called 'Chittron'. This model is trained to predict the caption when the input is an image, one word at a time.

Working Approach:

- The model has been trained on 15,700 images from the collected data set. Three hundred (300) images are considered as the test data.
- The model has two inputs: the first is the image itself, and the second is a sequence of tokens.
- After generating corresponding word embedding from tokens by embedding layer, these embedding form the sequence data input for the stacked LSTM layers.

- VGG16 model, slightly adjusted, has been used as the pre-trained image model.
- Stacked LSTM layers have been used in one-word-at-a-time strategy to predict caption.

Deb et al. [7] have addressed a standard approach for Bengali image caption generation through subsampling the machine-translated dataset.

Working Approach:

- Google Translate has been employed to adapt the translation process
- To overcome ambiguous words, including actual context understanding gap, a Bengali rule based stemmer has been used.
- Pre-trained Inception-ResNet and VGG-16 models have been used for images' feature extraction.
- FastText library's models have been utilized for pre-trained word embedding. Authors also have introduced a pre-compiled word embedding model to facilitate the word representation process.

Kamal et al. [6] have developed a system namely, 'TextMage' that is capable of understanding visual scenes that belong to the Bangladeshi geographical context and use its knowledge to represent what it understands in Bengali.

Working Approach:

- 'BanglaLekhaImageCaption,' a dataset previously developed and published by the same author of this paper, has been used to train the model.
- VGG16 has been utilized for image feature extraction
- CNN has provided necessary feature vector as the encoder
- LSTM has been used as a language model to create textual descriptions from given data images.

Jishan et al. [5] have proposed a deep neural network-based image captioning model to generate image description. It has been claiming to generate pertinent descriptions based on the modular complexities of an image.

- A full dataset has been classified by utilizing CNN and VGG16 highlights.
- The model has implemented Conv2D features with the Maxpooling 2D and ReLU activation function.
- It has been defined as 256 filters in the LSTM and set dropout value 0.2 to generate a caption for input image.

Pei et al. [17] proposed the Memory-Attended Recurrent Network (MARN) for video captioning,to over come the issue of not capturing multiple visual context information of a word appearing in more than one relevant videos.

- It comprises of three components: an encoder, a recurrent attention-based decoder, and an attended memory decoder.
- Due to its outstanding performance and relatively high cost-efficiency, the authors select ResNet-101 pretrained on imagenet as the 2D-feature extractor of the encoder. The ResNeXt-101 with 3D convolutions pre-trained on the Kinetics dataset is used to extract 3D features.
- As the backbone of the decoder, a recurrent neural network is used to generate the caption word by word due to its powerful ability to model the temporal information by the recurrent structure.
- To vastly improve the quality of the generated caption by the Attention-based Recurrent Decoder, an Attended Memory Decoder has been proposed as an assistant decoder.
- LSTM has been replaced by GRU as the Attended Memory Decoder.

Zhang et al. [18] proposed a video captioning framework which integrates with task-driven dynamic fusion replacing different static fusion methods and providing vast improvement in video captionig field.

- This paper conducts the first in-depth analysis of the weakness inherent in video captioning data-driven static fusion methods.
- According to model status, the proposed model can select various fusion patterns adaptively.

- VGG-19 and GoogLeNet-bu4k for visual features, C3D pretrained on Sports-1M video dataset for motion features extraction.LSTM is used as language model.
- These features and model status are inputs to TDDF unit. A dynamic visual input is provided by the TDDF unit for each iteration of the LSTM decoder.

A novel architecture, namely Semantically Sensible Video Captioning (SSVC), was proposed in this paper [19], which is simply a combination of two novel approaches - "stacked attention" and "spatial hard pull" - on top of a base video-to-text architecture to produce captions from video sequences to alter the process of context generation. A time-distributed fully connected layer which is comprises of double LSTM layers with stacked attention; followed by two consecutive bidirectional LSTM layers is used in the proposed process. The fully connected layer operates independently on each frame and then its output transfers to the LSTM layers that works as language model.

A brand new subject of collaborative image captioning with humans in the loop is discussed in this paper. In the interactive scenario, we have access to both the test image and a sequence of (incomplete) user-input sentences, unlike automatic image captioning, where a given test image is the sole input in the inference stage. The problem is formulated as Visually Conditioned Sentence Completion (VCSC). For VCSC, for image caption completion, asynchronous bidirectional decoding (ABD-Cap) [20] has been proposed.iCap with ABD-Cap as the core module, a web-based interactive image captioning framework has been developed, that can predict new text with respect to a user's live feedback. The feasibility of our ideas is shown by a variety of tests covering both automated assessments and individual user studies.

Chapter 3

Background Study

3.1 Convolutional Neural Network

In deep learning, Convolutional Neural Network (CNN) is an advanced feed-forward artificial neural network used for image classification and recognition tasks. It is an architecture similar to multi-layer perceptron, which works with supervised data and decides whether a particular value belongs to a class or not. But perceptron is not regularized because of data over-fitting. CNN regularizes data by adding wights to the loss function. CNN is a combination of neurons that have weights and biases, which can be changed based on the learning of the CNN model. Each neuron receives some inputs and executes a dot product, with non-linearity being an optional part. The layers are divided into three dimensions: height, width, and depth.
The basic architecture of CNN is a combination of different layers like input layer, convolution layer, pooling layer, fully connected layer and output layer which is shown in Figure 3.1.

- **Input Layer:** It is the very first layer of a CNN where the neurons take inputs and provide to the system for specific tasks. The weights and biases of the neurons in this layer are initiated randomly. All the neurons of this layer are connected to the neurons in the next layer.

- **Convolution Layer:** It is a core component of the CNN which does most of the works. This layer contains a set of small filters or matrices to be more specific, which slides through the height and width of the input volume. All the convolution layer executes convolution operation, which is a two-function mathematical operation (f and g), and it generates a third function (Equation 3.1). The convolution operation of f and g is denoted as f*g. After one is reversed and moved, it is known as an integral part of the two functions' product. This operation is a specific type of integral transforma-

tion [21].

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{3.1}$$

- **Pooling Layer:** This layer is usually used after the convolution layer. This layer executes a different kind of pooling operations and reduces the size of the input in order to minimize parameters as well as the computational steps. The pooling task is done by calculating the summary of a region of fixed size. The summarizing is done by taking maximum value, minimum value, an average value or by means of any mathematical calculation. Depending on the method of summarizing, there are different pooling functions like Max Pooling, Min Pooling, Average Pooling and so on.

- **Fully Connected Layer:** Like the neural network, every neuron in this layer is connected to the neurons of its previous layer. Its activation is also computed by matrix multiplication with its weight followed by bias as like neural network. Usually, a fully connected layer forms a column vector of the same size as the output class.

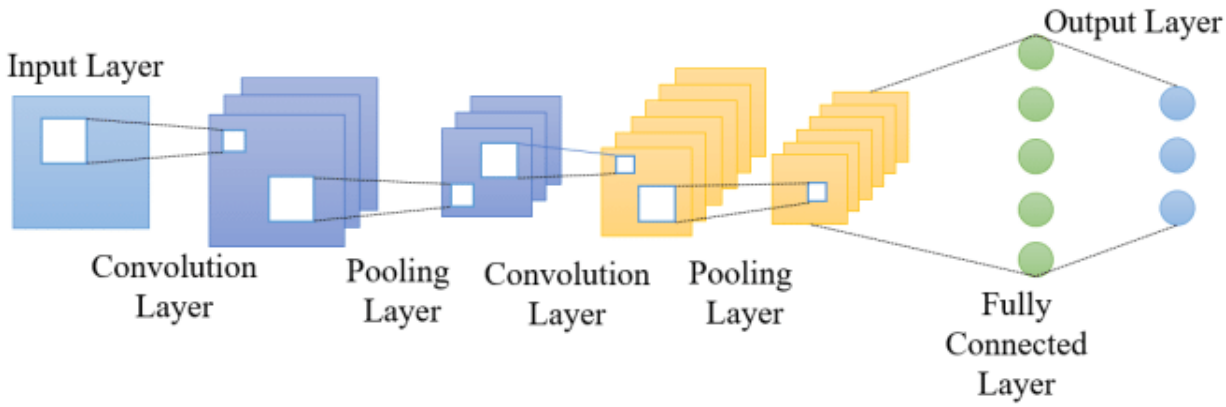


Figure 3.1: Basic Architecture of Convolutional Neural Network (CNN) [22]

3.1.1 2D CNN

When it is talked about Convolutional Neural Network (CNN), it is generally referred to as two dimensional CNN, mainly used on image data. This type of CNN is known as 2D CNN or Conv2D. Based on the dimension of the utilized convolutional kernal, CNNs can be categorized into three, and 2D CNN is one of them. Since 2D CNNs use 2D convolutional kernels that slide along two dimensions of the data, it is called two dimensional CNN. A basic model architecture is depicted in Figure 3.2.

A 2D convolution layer means that the convolution operation's input is three-dimensional, such as a color picture with a value across three layers for each pixel: red, blue and green.

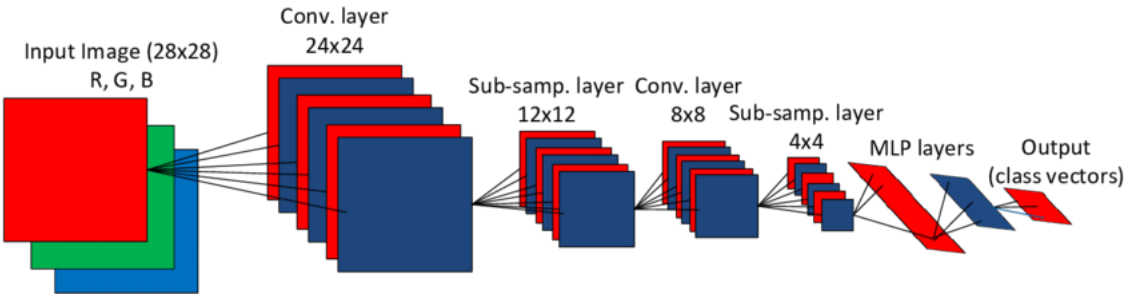


Figure 3.2: Model architecture of a standard 2D CNN. [23]

However, it is called a "2D convolution" since the filter's movement through the image occurs in two dimensions. Three times, once for each of the three layers, the filter is run across the image. The full benefit of using CNN is that it can use its kernel to extract spatial features from the data, which other networks can not do. For example, in the image, CNN can detect edges, color distribution, etc., making these networks very robust in image classification and other related data containing spatial properties.

3.1.2 3D CNN

Another type of CNN is three dimensional CNN, usually known as 3D CNN or Conv3D. In order to improve the identification of moving and 3D images, 3D CNNs are created, especially the 3D activation map constructed during the convolution of a 3D CNN is essential to analyze data where volumetric context is significant. Moreover, the time dimension is also very crucial. This third dimension is time in videos, which are as like as several pictures stacked together. Again, the height or number of layers, such as the layered image structure of an MRI scan, may also be used. In both cases, the third axis binds the two-dimensional sections intrinsically together and can thus not be ignored. 3D CNN tackles this case.
A 3-dimensional filter is applied to the dataset by 3D convolutions and the filter moves in 3-direction (x, y, z) to measure the representations of the low level feature. A 3-dimensional volume space such as a cube or cuboid is their output form. They are useful in the identification of events in videos, 3D medical images, etc. They are not limited to 3d space, but can also be applied, such as images, to 2d space inputs.

3.1.3 VGG 19 Model

VGG-19 is a modified version of VGG Net, a pre-trained CNN model first introduced by Simonyan and Zisserman from Visual Geometry Group (VGG) [24]. VGG-19 has been trained on ImageNet dataset [25]. This VGG variant consists of 19 layers, mainly including 16 convolution (Conv) layers and three fully connected (FC) layers. It also has five max-pooling

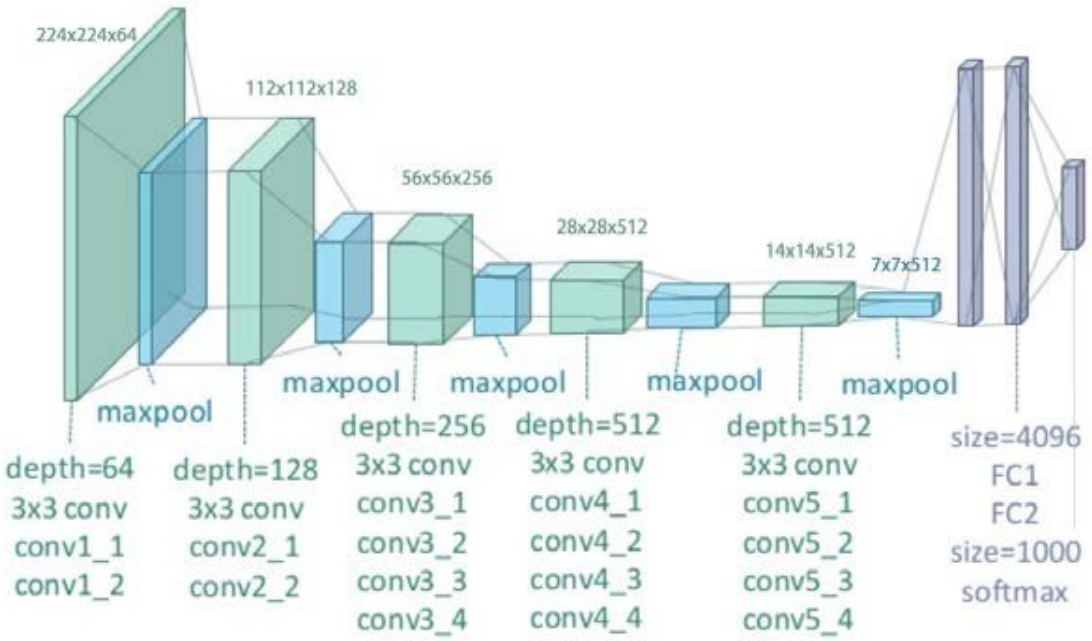


Figure 3.3: Basic architecture of VGG-19 model. [26]

layers and one softmax layers. All the layers are shown in Figure 3.3. The overall architecture is described here:

- This model takes the image of size 224x224 are used as input. The first two layers (convolution layers) use 64 filters with window-size of 3x3 and stride of 1 (Figure 3.3). So, the output size from the 2nd layer becomes 224x224x64.
- 3rd layer is a pooling layer which reduces the size down to 112x112x64.
- 4th and 5th layers are again convolution layers with 128 filters with the same window-size as previous layers. Now the output dimension will be 112x112x128. 6th layer is a pooling layer reducing the dimension to 56x56x128.
- 7th, 8th, and 9th layers are two convolution layers with 256 filters followed by a pooling layer.
- Layers from 10th to 19th have eight convolution layers of 512 filters and two pooling layers.The 19th layer outputs vector of dimension 7x7x512 (Figure 3.3).
- The input size for fully connected (FC) layer with 4096 units is 7x7x512 which is converted into a 1000 dimensional vector representing probability value for 1000 classes.

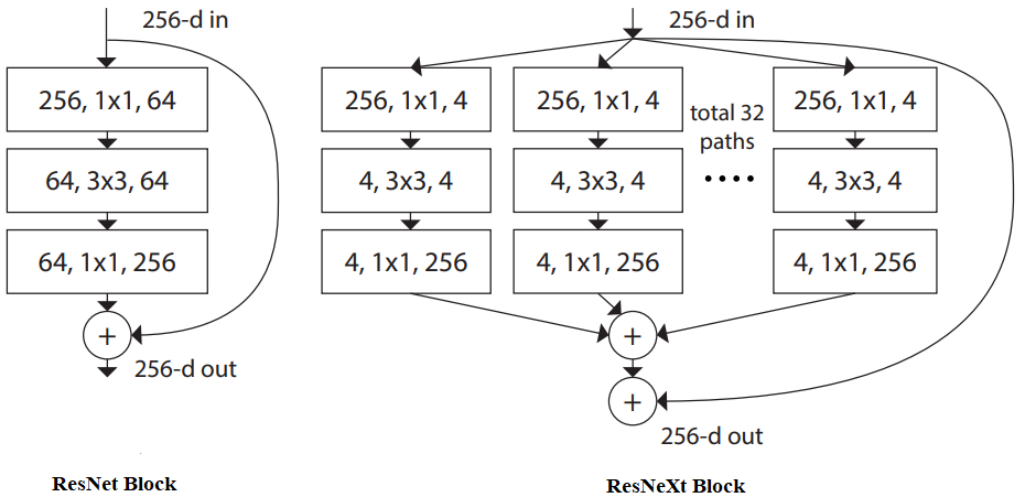


Figure 3.4: Side by side comparison of a single layer ResNet Block (Left) and a single layer ResNeXt block (Right). [28]

3.1.4 ResNeXt-101 Model

The ResNeXt architecture is a modified version of a CNN based model, Deep Residual Network (ResNet) [27]. ResNeXt-101 model is trained on ImageNet-5K dataset [28]. Explicitly, ResNeXt-101 model follows a three-step process- split, transform, and merge. In deep neural networks, the computational cost is very high, which is why ResNeXt-101 models represent the full input feature into some lower dimension of input block similar to ResNet blocks (Figure 3.4) and perform convolution which performs better and faster than executing convolution on the big input feature like all other neural networks. In the end, all the results are merged together. The detailed architecture(Figure 3.5) of a whole ResNeXt-101 model is here:

- First convolution layer (Conv1) has 64 filters with window-size of 7x7 and stride of 2. This layer output vector with dimension 112x112.
- Next layer is a max-pooling layer of size 3x3 and stride 2. The pooling layer transfers the output to three consecutive ResNeXt Block known as Conv2 layers shown in Figure 3.4 with the cardinality of 32. The output of the conv2 layer has 56x56 dimensions.
- The output of the conv2 layer goes through conv3 layer consisting of four ResNeXt Blocks. This process goes on upto conv5 layer. The conv5 layer outputs vectors of dimension 7x7.
- The 7x7 vector goes through the average pooling layer.
- The output from the average pooling layer goes as input of fully connected (FC) layer of 1000 dimension, and the output is 1x1 vector.

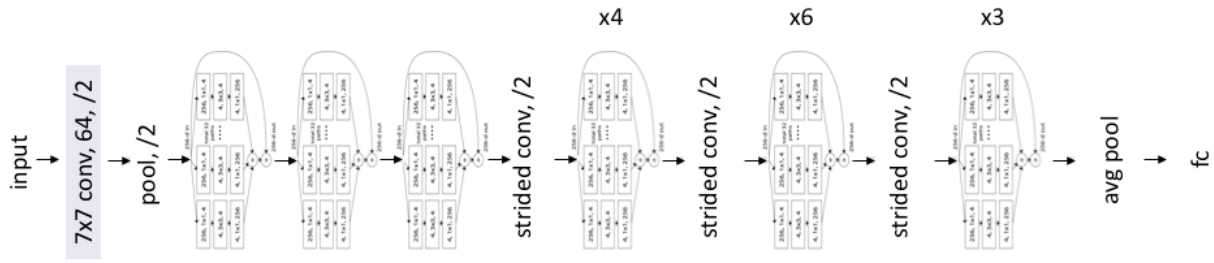


Figure 3.5: Full architecture of ResNeXt model. [29]

3.2 Recurrent Neural Network

Recurrent Neural Network (RNN) is another type of Neural Network that takes decisions not only based on the current step but also the previous step. Both the inputs and outputs are independent of each other in traditional neural networks, but in cases such as when it is vital to predicting the next word of a sentence, the previous words are needed, and so the previous words need to be remembered. RNN thus came into use, which with the support of a Hidden Layer, solved this problem. Hidden State, which recalls some details about a sequence, is the main and most significant feature of RNN. A basic structure of RNN is shown in Figure 3.6

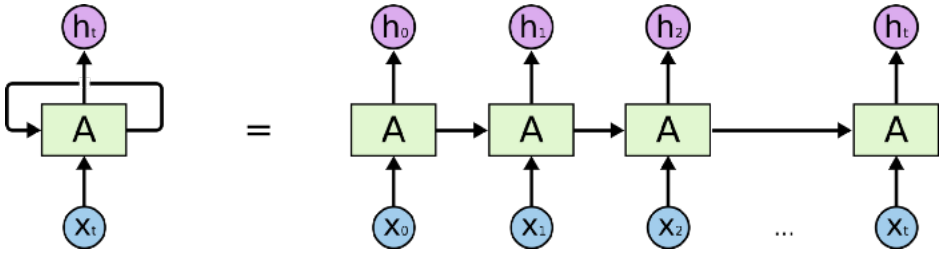


Figure 3.6: Basic Architecture of Recurrent Neural Network (RNN) [30]

By giving all layers the same weights and biases, RNN transforms the independent activations into dependent activations, reducing the difficulty of increasing parameters and memorizing each previous output by giving each output to the next hidden layer as an input. Therefore all three layers can be merged into a single recurrent layer such that the weights and biases of all the hidden layers are the same. Because of the training through these hidden layers, the RNN performs so fascinatingly in the case of sequential data. A simple pictorial representation of these training steps among recurrent layers is shown in Figure 3.7

The network is provided with a single time step for the input. Then the current input set is used and the previous state to determine the current state. For the next time step, the current ht becomes ht-1. According to the problem, one can go as many time steps and join the previous states' data. Upon completion of all time steps, the final current state is used

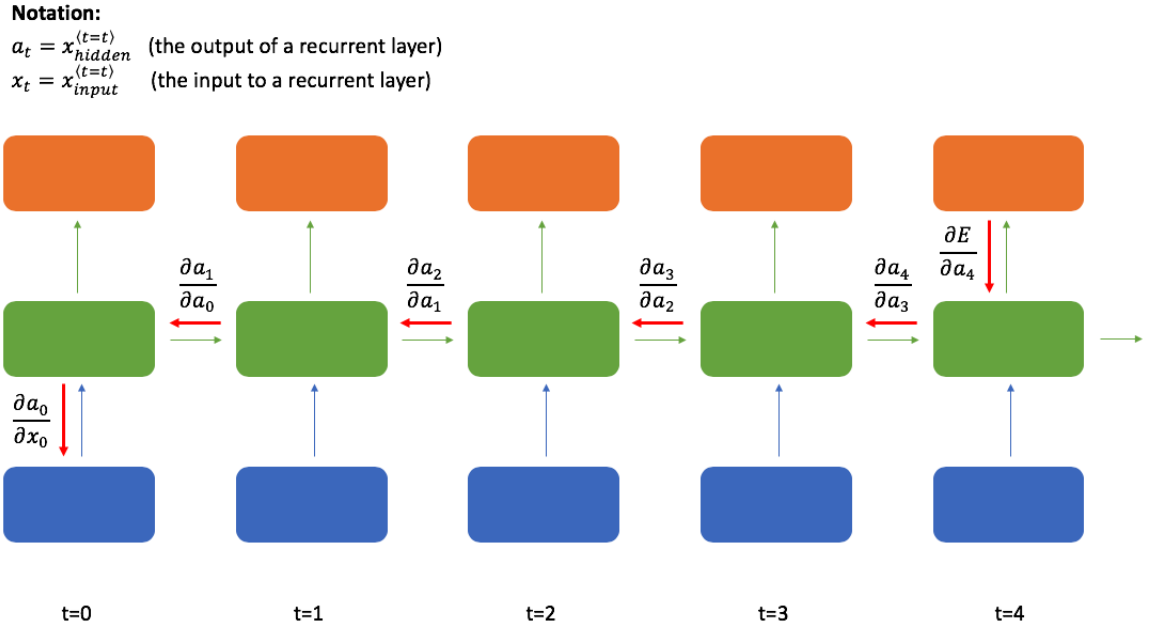


Figure 3.7: Backpropagation steps among Recurrent Layers [31]

to measure the output. Compared to the real output, i.e., the target output, the output is then produced, and the error is generated.

3.2.1 Long Shot-Term Memory (LSTM)

Long Short-Term memory is a special kind of RNN which resolves the lacking of long term dependency problems. Which means, this model is capable of learning long dependencies allowing it to learn long information. For example, previous state-of-the art methods of neural networks struggled to learn for connecting the recent past information and the present task if the gap between these two grows. But LSTM can perfectly handle the situation [32]. For this reason, LSTM is widely used as language model.

Like regular RNN, LSTMs also have a chain-like structure, but the repeating module chain has a different structure. There are four layers instead of a single neural network layer, interacting uniquely. The cell state, a horizontal line running down the entire chain, with only some slight linear interactions, is the key to LSTMs (Figure 3.8). It's effortless for data to only flow unchanged across it.

The LSTM has the capacity, carefully controlled by structures called gates, to remove or add information to the cell state. Gates are an optional way of allowing data to move through. An LSTM has three types of gates; namely, Input Gate, Forget Gate, Output Gate.

- **Forget Gate:** A forget gate (Figure 3.9) is responsible for pulling out details from

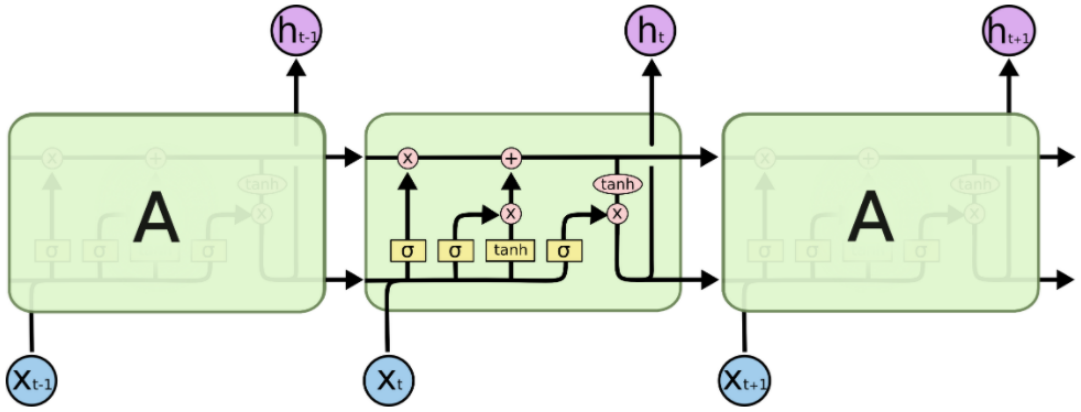


Figure 3.8: Basic Architecture of Long Short-Term Memory (LSTM) [33]

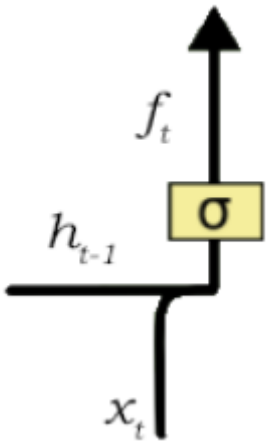


Figure 3.9: Forget gate of LSTM [34]

the cell state. By multiplying a filter, the information that is no longer needed for the LSTM to understand things or information that is of less value is removed. This is required for the efficiency of the LSTM network to be optimized. As depicted in the Equation 3.2, this gate takes in two inputs; h_{t-1} and x_t. h_{t-1} is the hidden state from the previous cell, or the output of the previous cell and x_t is the input at that particular time step.

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f) \tag{3.2}$$

- **Input Gate:** For the addition of information to the cell state, the input gate (Figure 3.10) is responsible. This data addition is essentially a three-step operation. Firstly, by involving a sigmoid function, controlling what values need to be added to the cell state. As shown in the Equation 3.3 and Equation 3.4, this is very similar to the forget gate and acts as a filter for all h_{t-1} and x_t data. Secondly, creating a vector that includes all possible values that can be added to the cell state. This is achieved using the tanh function, which outputs values ranges between -1 and 1. And finally,

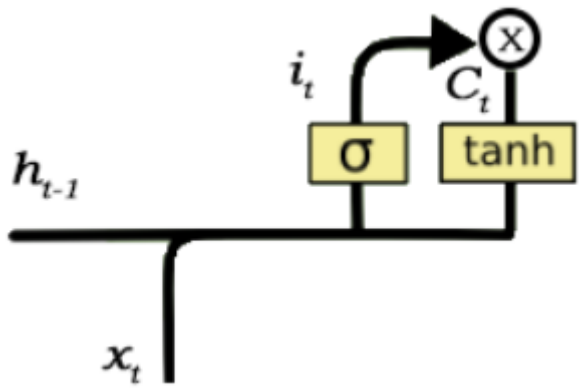


Figure 3.10: Input gate of LSTM [34]

multiply the value of the regulatory filter (sigmoid gate) to the generated vector (tanh function) and then through the addition operation, add this useful information to the cell state.

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i) \tag{3.3}$$

$$\widetilde{C}_t = \tanh(W_c * [h_{t-1}, x_t] + b_c) \tag{3.4}$$

- **Output Gate:** The output gate performs the job of selecting useful information from the current cell state and presenting it as an output. The activity of an output gate (Figure 3.11) can be split into three steps again and the required two equations are 3.5 and 3.6. First of all, after applying the tanh function to the cell state, constructing a vector, and scaling the values to the range of -1 to +1. Next, developing a filter using the h_{t-1} and x_t values so that the values that need to be output from the vector generated above can be managed. Once again, this filter employs a sigmoid function. Thirdly, multiplying the value of this regulatory filter to the vector generated in step 1, and sending it out to the next cell as an output, as well as to the hidden state. The

$$o_t = \sigma(W_i * [h_{t-1}, x_t] + b_o) \tag{3.5}$$

$$h_t = o_t * \tanh(C_t) \tag{3.6}$$

3.2.2 Bidirectional LSTM

A Bidirectional LSTM, or biLSTM, is a model of sequence processing consisting of two LSTMs: one in the forward direction of the input and the other in the backward order.

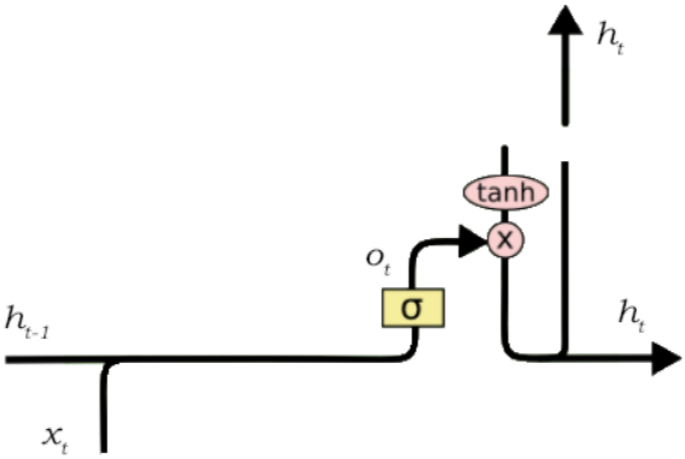


Figure 3.11: Output gate of LSTM [34]

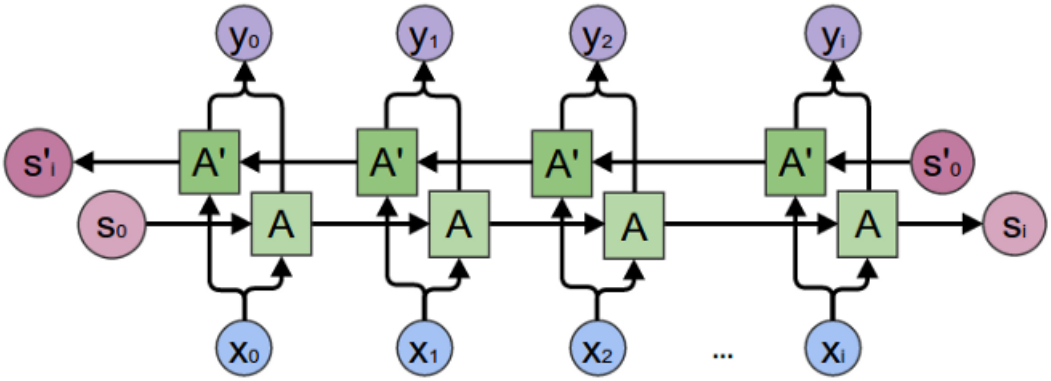


Figure 3.12: Basic Architecture of Bidirectional LSTM [30]

BiLSTM looks precisely the same as its unidirectional counterpart. The main difference lies in its purpose. The Bi-LSTM aims to look at a specific sequence from both front-to-back and back-to-front. Consequently, it effectively increased the amount of information available to the network, improving the context available to the algorithm. In this way, especially in the case of the language model, for each character in the text, the network generates a meaning that depends both on its past and its future. BiLSTM has three gates in total, from which, two of them are similar to unidirectional LSTM, and a new one namely update gate, and its Equation is 3.7.

$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u) \tag{3.7}$$

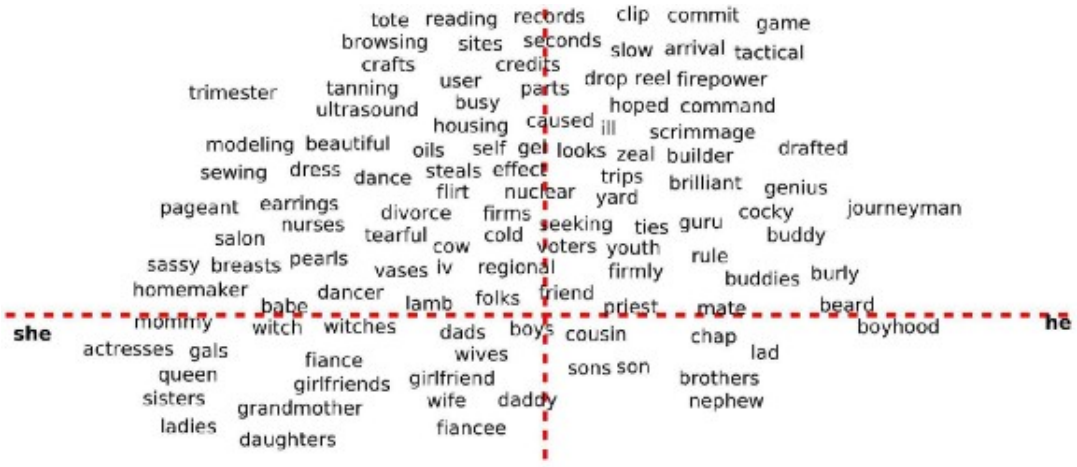


Figure 3.13: Example of word embedding: similar word being close to each other. [35]

3.3 Word Embedding

Word embedding is a crucial component for accurate results in the field of Natural Language Processing (NLP). A language model needs to learn similar words based on similar contexts which would allow the model to generate proper and meaningful sentences using near correct words. Word embedding enables a model to do so by assigning numbers to each word in a way that similar words are in the same clusters and other words are at different distances based on the relations between the words (Figure 3.13).
Machine Learning (ML) or Deep Learning (DL) models are mostly unable to take raw text data as input because the neurons in deep leaning or the machine learning layers deal with numbers and outputs some numbers. In the simplest form word embedding, a word is represented with a vector called "One-hot Vector". If the vocabulary size is N, the one-hot vector will have length N, and according to the position of a word, only one value of the vector will be one and rest of them will be zero.

3.3.1 Word2Vec

Word to Vector or Word2Vec is a trendy word embedding approach in Natural Language Processing. It was developed by a team in Google, led by Tomas Mikolov [36]. The task of Word2Vec is achieved through two models: Continuous Bag-of-Words Model (CBOW) and Continuous Skip Gram .

- **Continuous Bag-of-Words Model:** The input of the CBOW model is the context of each word in the vocabulary, and depending on that; it tries to predict the corresponding word according to the context [36]. CBOW represents the occurrence of words in a text document. This model discards any information related to the structure or

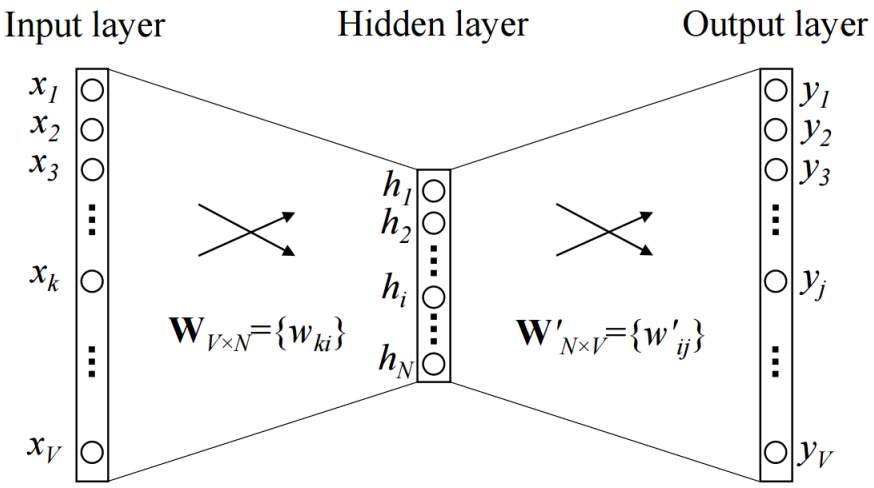


Figure 3.14: Basic architecture of Continuous Bag-of-Words (CBOW) model with one context word. [35]

order of any word and deals with whether a word occurs in the document or not. We can have a good of the basic architecture of CBOW model from Figure 3.14. Let's assume, a context or document has V. If a single word is given to predict next word, the model will have a one-hot encoded vector of length V. The hidden layer will have any number of neurons, $N < V$ and the output layer will again be a vector of V length having all the softmax values.

- **Continuous Skip Gram Model:** This model is very much similar to the architecture of CBOW model. Continuous Skip Gram model is one of the unsupervised learning methods which predicts the words surrounding a given input word where CBOW predicts only the next word. This model executes all the works with a view to maximizing the classification of a word depending on other words in the sentence [36]. The dot product of the input word $w(t)$ and the weight matrix in the hidden layer or projection layer is calculated which is the output of projection layer. The layers are shown in Figure 3.15. The projection layer has no activation function. The output of the projection layer is transferred to the output layer where the output layer again computes the dot product of the output of the projection layer and the wight matrix in the output layer. At last, softmax of all the values are computed to get the word prediction.

3.3.2 FastText

FastText [37] is a lightweight library created by Facebook's AI Research (FAIR) lab aiming for efficient learning for text representation as well as text classification. The architecture consists of three layers, input layer, hidden layer, and output layer as shown in the Figure 3.16. It is written in C++ and allows both supervised and unsupervised learning for obtain-

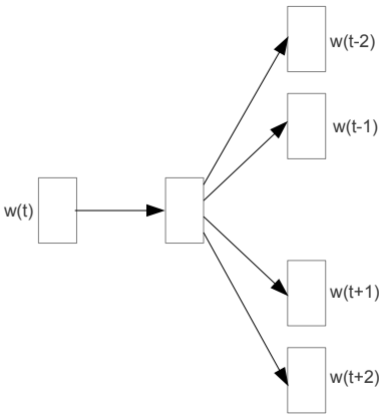


Figure 3.15: Basic working principle of Continuous Skip Gram model. [36]

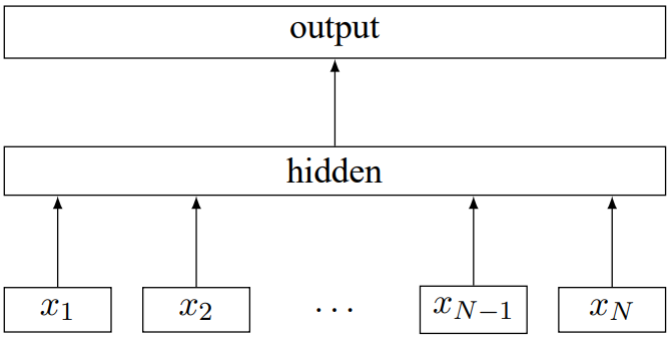


Figure 3.16: Model architecture of FastText for N n-gram features. [38]

ing word embeddings. FastText uses negative sampling, softmax or hierarchical softmax loss functions to enable the training of continuous Bag-of-Words (CBOW) or Skip-gram models. FastText consists of pre-trained models that are trained on Wikipedia and over 150 different languages. To overcome the limitations of Word2Vec, such as Out of Vocabulary (OOV) words, Morphology etc., [37] have proposed FastText. The FastText algorithm is also vastly inspired by [38].

3.3.3 GloVe

GloVe [39] stands for "Global Vectors". GloVe is an algorithm for unsupervised learning to obtain vector representations for words. To come up with word vectors, GloVe gathers both global statistics and local statistics of a corpus. Training is carried out on these aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations show interesting linear substructures of the word vector space. Generally, the GloVe algorithm can be splited into the following three tasks:

- Representing word co-occurrence statistics in the form of a word co-occurrence matrix, named **X**. Each element, X_{ij} of this matrix, **X** represents how often word i appears in

context of word j. The corpus should be scanned in the following manner: for each term, context terms are being looked for within some area defined by a *window-size* before the term and a *window-size* after the term. Also, more distant words are being assigned less weight using the following Equation 3.8;

$$decay = 1/offset \tag{3.8}$$

- For each word pair soft constraints must be defined using the following Equation 3.9

$$w_i^T + b_i + b_j = log(X_{ij}) \tag{3.9}$$

- A cost function need to be defined like the Equation 3.10.

$$J = \sum_{i=1}^{V} \sum_{j=1}^{V} f(X_{ij})(w_i^T w_j + b_i + b_j - logX_{ij})^2 \tag{3.10}$$

Chapter 4

Proposed Methodology

4.1 Overview

To generate Bengali captions from videos, we have used different deep learning models and architectures. For extracting both spatial and temporal features from the videos, we have used both 2D and 3D convolutional neural network (CNN) models such as VGG-19 and ResNeXt-101, respectively. For embedding the words of the captions, we have used three embedding methods, namely, FastText, Glove and Word2Vec. In all these models, we have leveraged the advantage provided by transfer learning by using pre-trained models using large datasets. In the encoder and decoder, we have used the sequential model such as Bidirectional Long Short-Term Memory (Bi-LSTM) and two-layer LSTM, respectively, to generate the captions (Figure 4.1).

4.2 Pre-processing

Pre-processing is an important step to make the data contained in the dataset understandable by the learning architecture. It is the step before the data are being fed into the model. Pre-processing discards the unnecessary information contained in the dataset and extracts only the useful information needed for the model.
The dataset consists of two kinds of data. They are - the videos and their respective captions. To make the videos and captions useful for feature extraction and embedding respectively pre-processing was performed. The pre-processing steps are described in the following sections.

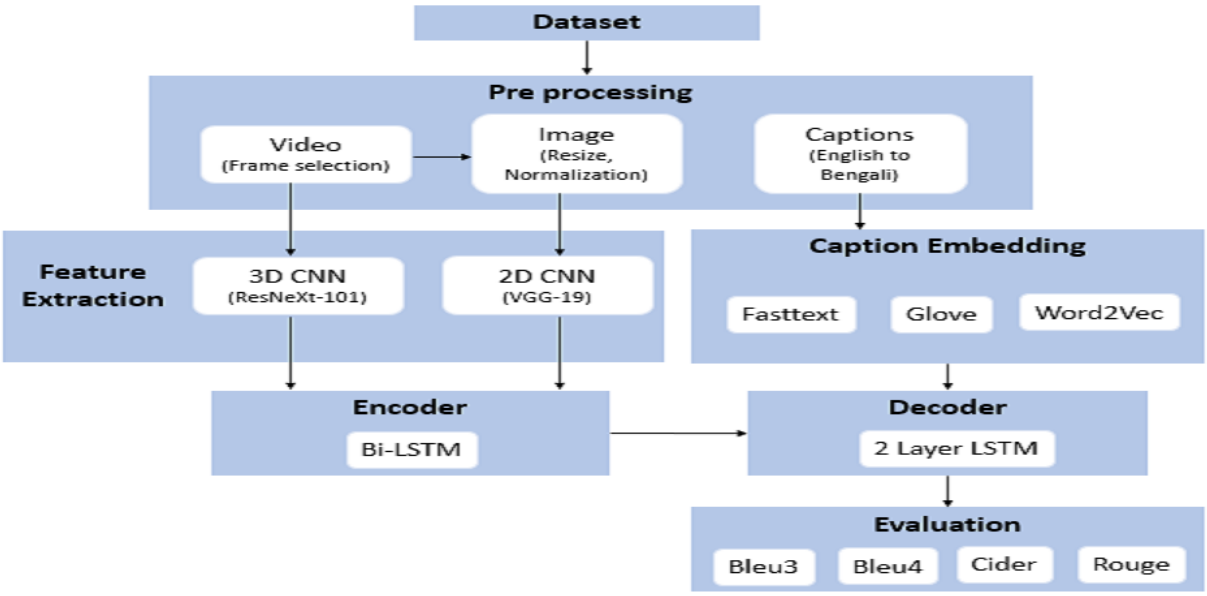


Figure 4.1: Proposed Methodology for generating Bengali captions from the videos.

4.2.1 Video Pre-processing

The videos of the dataset have 30 frames per second (fps). To make the model efficient and avoid redundancy, we have selected three frames per second on the basis of sampling video frames once in every ten frames. After this first sampling, the average frame number per video on the dataset was 31 frames. So, to make all the videos having the same number of frames as per the requirement of the feature extraction model, we have taken 32 frames from all the videos by applying the process replication and truncation of frames.

4.2.2 Image Pre-processing

All the frames extracted from the videos are needed to be of uniform dimension for them to be fed in the feature extraction model. The frames were resized to the dimension of 224x224x3 as per the requirement of the VGG-19 feature extraction model.
Since the feature extraction was done using VGG-19 architecture pre-trained on ImageNet dataset, normalization was performed using the mean and standard deviation (std) calculated from millions of images of ImageNet dataset. The images were first loaded in to a range of [0,1] and then normalized using mean = [0.485,0.456,0.406] and std = [0.229,0.224,0.225].
Here, the mean is the sequence of means for each of the three channels (red, green, blue) and likewise the std is the sequence of standard deviations for each of the three channels. Normalization for each channel is calculated by equation 4.1,

$$input_{(channel))} = (input_{(channel))} - mean_{(channel))})/std_{(channel))} \tag{4.1}$$

Normalization helps the CNN model perform better and gets the data within a range and reduces the skewness of the data.

4.2.3 Caption Pre-processing

The dataset contains 85,550 English captions, averaging about 43 captions per video. All of these captions were translated into Bengali using Google Translate API in Python. Due to the limitations of the API, some of the captions had unwanted noise like containing untranslated English words on the Bengali caption.
These unwanted noises and also the punctuation marks and other signs were removed from the Bengali captions. The sentences were then tokenized to create a vocabulary containing only unique words. Special token like <start>, <end> were added to mark the beginning and end of the sentence. Token <pad> was included to make all the captions of uniform length, and the token <unk> was added to detect out of vocabulary representations. The final vocabulary contains 7,105 unique words after discarding rare words which were used less than three times in the dataset.

4.3 Feature Extraction

Feature extraction is a part of the dimensionality reduction process. When an input data to an algorithm or model is too large to be processed, or it is assumed to be redundant, then it is generally modified into a reduced set of features commonly known as a feature vector. An essential characteristic of a large dataset is that it has a large number of variables which require a lot of computational resources to process them. So, feature extraction helps to acquire and combine the best features from the dataset. It thus effectively reduces the amount of data but still describe the actual data accurately with originality. The reduction of data also helps in building the model with less computational effort and at the same time, increases the speed of learning and generalization steps in the machine learning process.
In our learning model, we have utilized both image and video features. The process of extracting features from both the images and videos are explained in the later sections.

4.3.1 Image Feature Extraction

Convolutional neural networks are widely used and it is very useful for extracting spatial features from each of the images taken from the videos. For feature extraction from the frames, we have used the 19-layer VGG convolutional neural network model. The model was pre-trained on ImageNet [25] dataset.

VGG-19 model takes 224x224x3 resized RGB image as input. In convolution layers, the model uses kernels of size (3 x 3) with a stride of 1 pixel that covers the whole notion of the image. Convolution layers are followed by Rectified Linear Unit (ReLU) activation to introduce non-linearity into the model. In between the convolutional layers, there are pooling layers which use max pooling over a (2 x 2) pixel window with a stride of 2. After the convolutional layers, there are three fully connected layers which are named as fc6, fc7 and fc8. The first two - fc6 and fc7 have the same dimension of 4096 units. The last fully connected layer has a dimension of 1000 units.
In our proposed methodology, we took the output of 4096-way fc7 fully connected layer from VGG-19 model and discarded the last fully connected layer. The last fully-connected layer with 1000 channels is for 1000-way classification which is followed by a softmax output of one of the 1000 classes. That is why we discarded the last fully connected layer.
After extracting spatial features from the frames of a video, the dimension of the features per video becomes (32 x 4096).

4.3.2 Video Feature Extraction

As the videos contain temporal information, only spatial description is not enough to produce good captions. So, to generate meaningful captions, both visual appearance and temporal information should be leveraged jointly. To extract temporal motion features, we have employed ResNeXt-101 implementation as the 3D-CNN based feature extraction model. The model was trained on Kinetics [40] dataset, which includes classes of 400 actions.
The activations of the last convolutional layer (2048 dimensions) of ResNeXt-101 are extracted as the temporal feature representation for every 16 frames of a video. For any video longer than 16 frames, they were segmented into non-overlapping 16-frame video snippets, and their extracted features were combined using max pooling. Here also we have discarded the last fully connected layer with softmax output used for classification.
After extracting temporal features from the videos, the dimension of the features per video becomes (1 x 2048).

4.3.3 Caption Embedding

The captions were tokenized, and a vocabulary of unique words was created in the preprocessing step. These words are represented in the feature space using word embeddings. Word embeddings are a set of techniques that represents individual words as real-valued vectors in a predefined vector space. The representation is learned based on the usage of words. It results in the words which share a lot of similar features to have similar representations.

In our proposed methodology, we have used three different word embedding methods. They are – Word2Vec, FastText and Glove. We have used the implementation and pre-trained model given by Bengali Natural Language Processing or BNLP toolkit. The models were trained with Bengali Wikipedia Dump Dataset [41].
After embedding, each word is represented as a 300-dimension feature vector.

4.4 Encoder-Decoder Model

To generate captions from the videos our proposed encoder-decoder framework is composed of two steps:

1. In encoder, the spatial and temporal features are encoded and combined using Bidirectional Long Short-Term Memory (Bi-LSTM) to generate a context vector.
2. In decoder, the context vector is fed to a two-layer LSTM network that generates the desired captions.

The steps are comprehensively described in the following sections and the Figure 4.2 shows the working principle of encoder-decoder model.

4.4.1 Encoder

To encode CNN features extracted from the video frames, Bi-LSTM is used in the encoder. Bi-LSTM duplicates the first recurrent layer in the network so that there are two layers now on side-by-side. The extracted features from the frames of videos are fed from beginning to end as input to the first layer and again from end to beginning to the second layer.
Using Bi-LSTM preserves the context across the frames of a video as at each time step within the bidirectional architecture, it is possible to see not only the past frames but also take a look at the future frames. So, the Bi-LSTM architecture encodes the video by scanning through the whole video several times where each scan is relevant to its subsequent scan.
The output vector of the last cell of the Bi-LSTM model and the extracted feature of the video by ResNeXt-101 model is then concatenated and passed through a linear layer to generate a unified context vector. The activation function used here is the Tanh activation function to keep the values of the context vector within a range of [-1,1]. To reduce the effect of overfitting, a dropout is also used in the encoder.

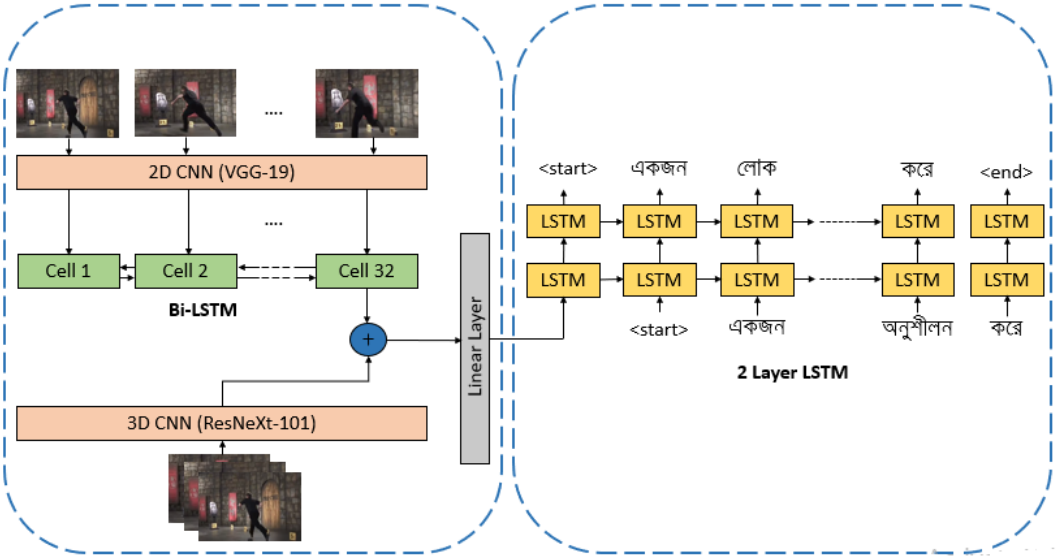


Figure 4.2: Proposed Encoder-Decoder model. In encoder, the extracted features from VGG-19 are encoded using Bi-LSTM and the output is later concatenated with the extracted features from ResNeXt-101 model. In decoder, two-layer LSTM is used which predicts the words in a sentence given the context vector learnt from the encoder and word embeddings.

4.4.2 Decoder

In the decoder, a two-layer LSTM is used as the language model. With sufficient training data, LSTM architectures can learn long term dependencies between events and generate sentences that look like natural sentences. As LSTM operates on sequential data, adding a second LSTM layer on top of the first layer adds levels of abstraction of input sequences over time. Additional hidden layers of the stacked LSTM layer recombine the learned representations from the previous layer and create new high-level representations. This results in the overall robustness of the model.
The LSTMs predict the probabilistic distribution of the output caption given the input video context. Let the context vector is denoted by V. The dictionary of words or vocabulary is denoted by D. The output sentence can be denoted by the Equation 4.2.

$$S = \{w_0, w_1, w_2, ..., w_n\} where \;\; w_i \;\; \epsilon \;\; D \tag{4.2}$$

In the sentence, the specially added tokens <start> and <end> which is used for indicating the beginning and end of a sentence is denoted by and. The probability of the predicted words of a sentence can be expressed as the following Equation 4.3:

$$
\begin{aligned}
P(S|V) &= P(w_0, w_1, w_2, ..., w_n|V) \\
&= P(w_1, w_2, ..., w_n|V) \\
&= \prod_{t=1}^{n} p(w_t|h_{t-1}, l_t)
\end{aligned}
\tag{4.3}
$$

Since the beginning of the sentence is always <start>, the word prediction actually starts at time step t = 1. At each time step, the LSTM cell calculates the value of hidden layer h_t given the hidden layer value of previous time step $h_{(t-1)}$ and output of the layer below l_t.

4.5 Summary

In this chapter, the proposed methodology for Bengali caption generation from input videos is described with proper diagrams and explanations. Feature extraction and the proposed encoder-decoder model is thoroughly demonstrated here with a proper diagram and explanation. Detailed pre-processing tasks are highlighted in this chapter. The pre-processing includes selecting random 32 frames from each video, having an average length of 10 seconds. The video frames or images are resized to 224x224x3 dimensions for the feature extraction model, VGG-19. For the language generation part, all the captions are translated to Bengali from English using Google Translate API, and punctuation marks and noises are removed. 2D CNN-based VGG-19 model and 3D CNN-based ResNeXt-101 model are used for extracting both spatial and temporal features from the videos, respectively. For word embedding, three embedding methods, namely, FastText, Glove, and Word2Vec, have been used. In the encoder and decoder, the sequential model such as Bidirectional Long Short-Term Memory (Bi-LSTM) and two-layer LSTM are used to generate the captions.

Chapter 5

Experimental Results & Evaluation

5.1 Overview

In this section, we will describe the outcomes and behaviours of our proposed methodology extensively. We approached the video captioning problem as a sequence-to-sequence (seq2seq) modelling task and used RNNs in both encoder and decoder section. Following, we will evaluate the performances of all the experiments and compare them. Furthermore, as there is no pre-existing work on video captioning in Bengali, we will compare our model's performance with the existing models of image captioning in Bengali.

5.2 Experimental Setup

We used Jupyter Notebook and various Python packages such as Numpy, Pandas, OpenCV, H5py, bnltk etc. for image, video and language pre-processing. Image and video features were extracted from pre-trained models using torchvision.models, a built-in library in PyTorch. For training and testing our model, we used PyTorch framework. We used the dedicated GPU, which is provided by Google Colab.

5.3 Dataset Acquisition

For our work, we have used the Microsoft Video Definition (MSVD) dataset [42]. It contains 1970 videos from YouTube of average length of 10 seconds with sentences annotated by AMT staffs. The dataset has total of 70,028 sentences for 1970 video clips each having multiple reference captions.
All the captions in MSVD are annotated in English. We translated all the English captions

to Bengali using Google Translation API. The sentences after translation are not as good as the human annotations which effects the accuracy of the model.The dataset is divided into training, testing, and validation set having 1200, 670, and 100 video clips in the sets respectively.

5.4 Performance Measurement

5.4.1 BLEU

BLEU [43] is a famous metric, very first introduced in 2002, used to measure the quality of the text generated by the machine. The quality measures the similarity between human outputs and outputs generated by machines. According to BLEU, a high-scoring description should correspond in length to the ground-truth sentence such as the exact match of words as well as their order. BLEU scores take into consideration the similarity between predicted unigrams (single word) or higher n-gram and a set of reference sentences of one or more candidates. BLEU evaluation will show '1' as an output score when an exact match occurs. The Equation 5.1 represents the formula of BLEU.

$$\log BLEU = min(1 - \frac{l_r}{l_c}, 0) + \sum_{n=1}^{N} \omega_n \log p_n \tag{5.1}$$

In the above equation, $\frac{l_r}{l_c}$ is the ratio of the length of the analogous reference corpus, and the candidate description, ω_n are positive weights, and p_n indicates to the geometric average of the modified n-gram precisions.

5.4.2 CIDEr

CIDER [44] was first introduced in 2015 and mostly used for evaluating image caption quality. It evaluates the consensus of the corresponding image between a predicted sentence and the reference sentences. Each sentence is treated by CIDEr as a collection of n-grams containing 1 to 4 words. All the words in a sentence are first converted to their stem or root. Encode the consensus between the sentence expected and the reference sentence; it measures the frequency of n-grams co-existing in both sentences. For each n-gram, the weight is calculated using the Inverse Document Frequency Term Frequency (TF-IDF). Using

the Equation 5.2 CIDEr score can be computed.

$$CIDER_n(c_i, S_i) = \frac{1}{m}\sum_j \frac{(g^n(c_i)).(g^n(s_{ij}))}{||g^n(c_i)||.||g^n(s_{ij})||} \tag{5.2}$$

where $g^n(c_i)$ is a vector representing all n-grams with length n and $||g^n(c_i)||$ depicts the magnitude of $g^n(c_i)$. Same is also applicable for $g^n(s_{ij})$.

5.4.3 ROUGE

In 2004 ROUGE [45] metric was introduced to determine text summaries. Just like BLEU, ROUGE is also calculated by varying the count of the n-gram. Unlike precision-based BLEU, however, ROUGE is based on the recall values. It calculates the recall score of the generated sentences corresponding to the reference sentences using n-grams. There are different parts of ROUGE for different works. ROUGE-N measures overlap. It calculates uni-gram, bi-gram, tri-gram, and higher-order overlap. ROUGE-L measures the longest matching sequence of words in between two or more strings using LCS. ROUGE-S looks in a sentence for a pair of words in order. ROUGE-N is computed with the Equation 5.3.

$$ROGUE - N = \frac{\sum_{s \in R_{sum}} \sum_{g_n \in S} C_m(g_n)}{\sum_{s \in R_{sum}} \sum_{g_n \in S} C(g_n)} \tag{5.3}$$

Here, n being the n-gram length, g_n, and $C_m(g_n)$ represents the highest amount of n-grams that are present in the candidate as well as ground truth summaries and R_{Sum} stands for reference summaries.

5.5 Hyper-parameters setting

In this section, different setups based on hyper-parameter values are discussed. These information are depicted in the Table 5.1:

Table 5.1: Uniform Hyper-parameters for all the setups

Hyper-parameter	Value
Batch_size	500
Step_per_epoch	99
Epoch	50
Momentum	0.0

We experimented on four different setups by tuning the hyper-parameters, whose details are given in the Table 5.2:

Table 5.2: Hyper-parameter values for different setups

Setup	Encoder Hidden Size	Encoder Dropout	Decoder Hidden Size	Decoder Dropout	Word Embedding Dimension	Learning Rate
Setup-1	300	0.2	300	0.4	300	0.0005
Setup-2	500	0.2	500	0.4	500	0.005
Setup-3	1000	0.1	1000	0.5	1000	0.0002
Setup-4	1500	0.2	1500	0.4	1500	0.00005

5.6 Experimental Result

In this section, we will discuss the experimental result of the proposed methodology. We extracted Bengali word embedding from pre-trained FastText, Word2vec and Glove model. Each of them represents a word by a 300-dimensional vector. We expanded the dimension as a hyper-parameter by a linear layer and fine-tuned it. The following subsections describe the results thoroughly.

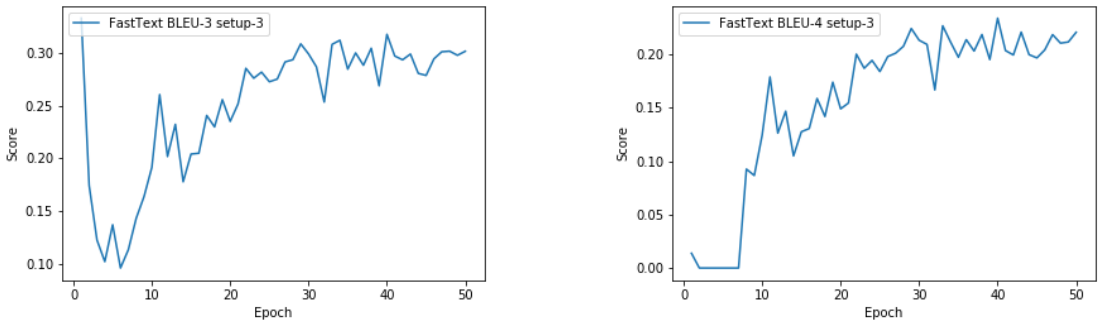
5.6.1 Experiments using FastText

In Table 5.3, we have shown the performance of four different setups while using the FastText model for word vectors. We found that, versions of BLEU metrics obtained better score than other metrics for setup-1. CIDEr and ROUGE worked better with setup-3 and showed better scores.

Table 5.3: Performance scores of models with FastText embedding on test data

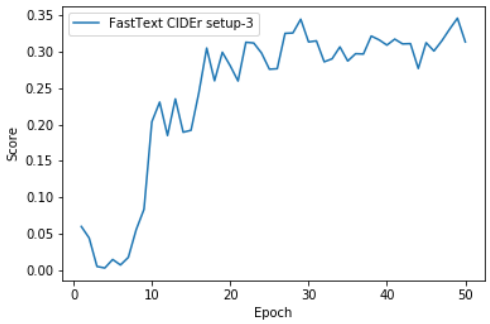
Setup	BLEU-3	BLEU-4	CIDEr	ROUGE
Setup-1	**0.321**	**0.223**	0.276	0.47
Setup-2	0.262	0.217	0.09	0.415
Setup-3	0.308	0.221	**0.324**	**0.496**
Setup-4	0.252	0.06	0.08	0.35

Figure 5.1 shows the validation set results for evaluation matrices over the iterations for setup-3 over 50 epochs. From the graphs, ROUGH stands out to be the highest validation accuracy achiever by far. We can see highest fluctuations in validation accuracy through out the 50 epochs.
In Figure 5.2, a representation of loss curve of the model for setup-3 using FastText embedding on training set is depicted.

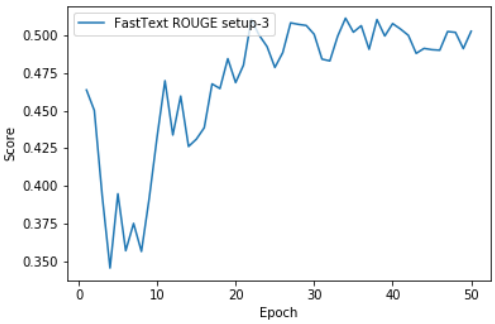


(a) Validation for BLEU-3

(b) Validation for BLEU-4



(c) Validation for CIDEr



(d) Validation for ROUGE

Figure 5.1: FastText Setup-3 Validation Accuracy Curves

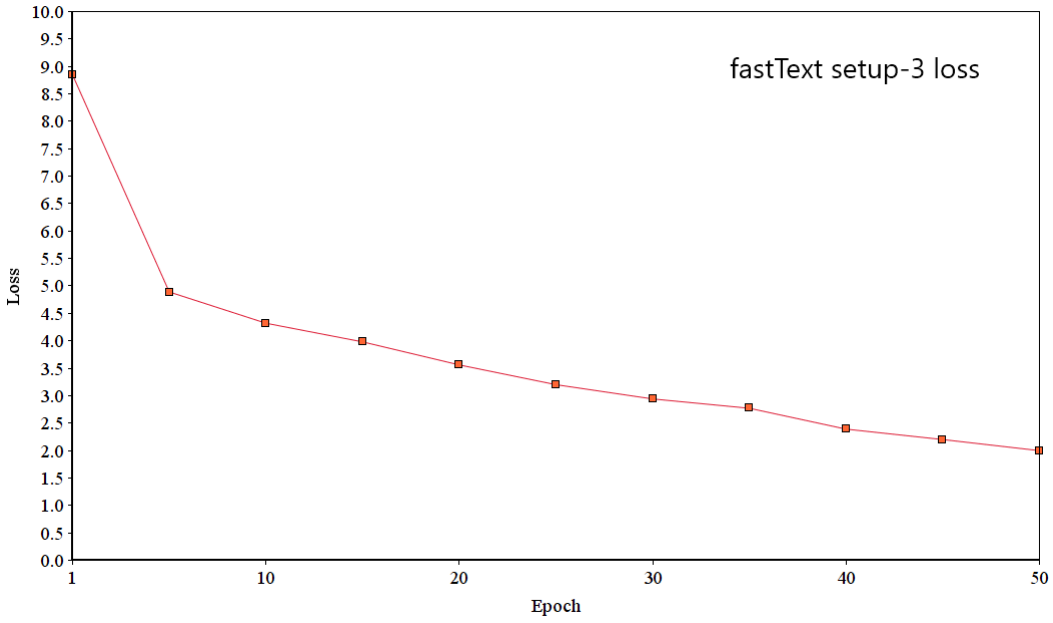


Figure 5.2: Model Loss Curve for FastText embedding with setup-3

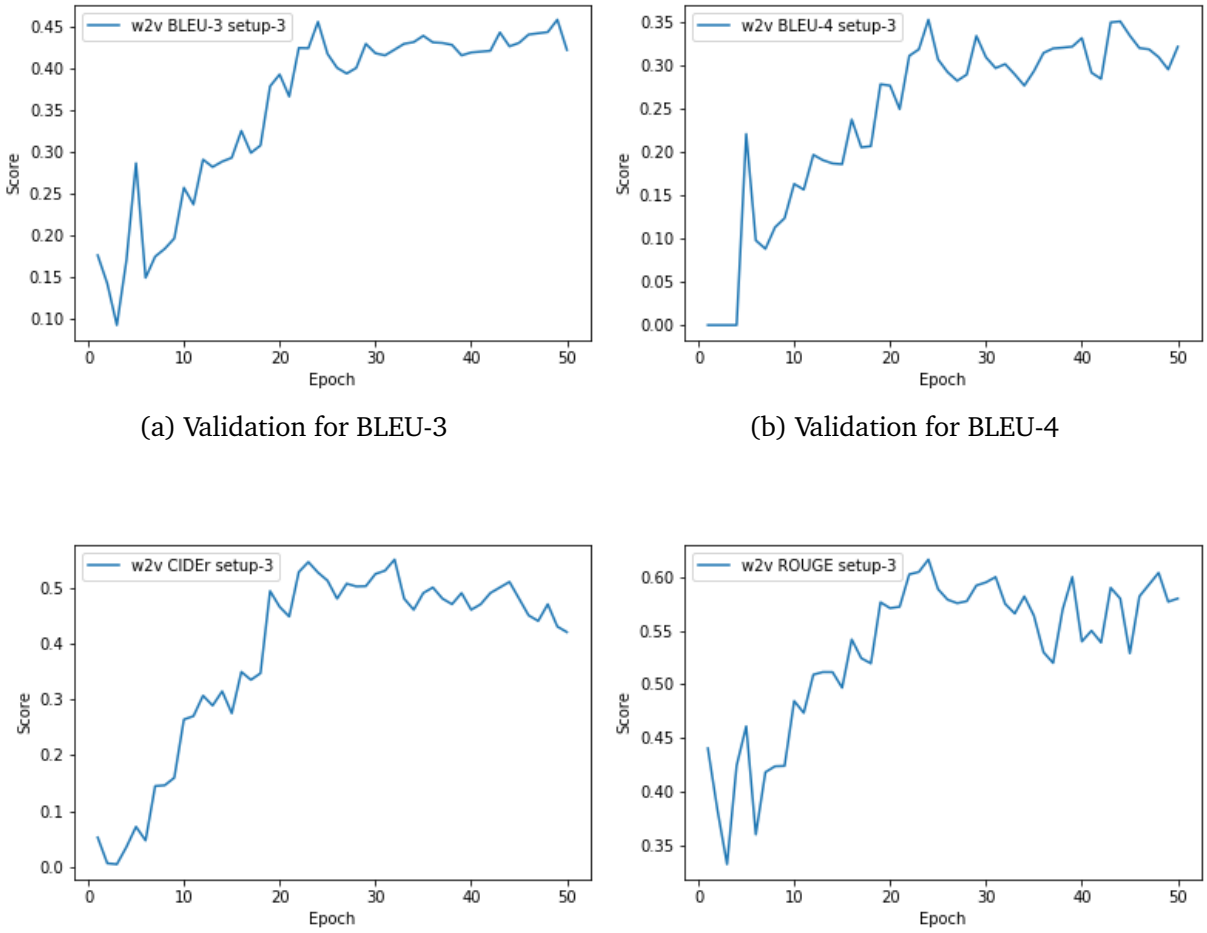
5.6.2 Experiments using Word2vec

In Table 5.4, we have shown the performance of four different setup while using the Word2vec model for word vectors. Setup-3 outperformed other setups and got the highest value in all four evaluation matrices.

Table 5.4: Performance scores of models with Word2vec embedding

Setup	BLEU-3	BLEU-4	CIDEr	ROUGE
Setup-1	0.286	0.220	0.314	0.502
Setup-2	0.273	0.231	0.112	0.438
Setup-3	**0.432**	**0.326**	**0.512**	**0.573**
Setup-4	0.288	0.185	0.276	0.493

Figure 5.3 shows the validation set results for evaluation matrices over the iterations for setup-3 using Word2vec embedding model.



(a) Validation for BLEU-3

(b) Validation for BLEU-4

(c) Validation for CIDEr

(d) Validation for ROUGE

Figure 5.3: Word2vec Setup-3 Validation Accuracy Curves

In Figure 5.4, a representation of loss curve of the model for setup-3 using Word2vec embedding on training set is depicted.

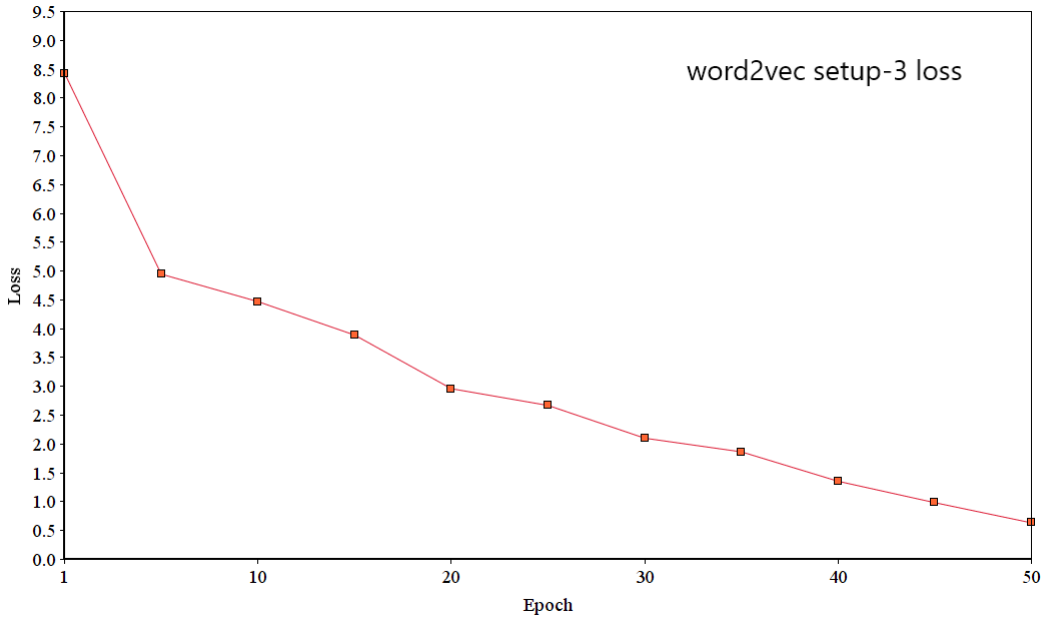


Figure 5.4: Model Loss Curve for Word2vec embedding with setup-3

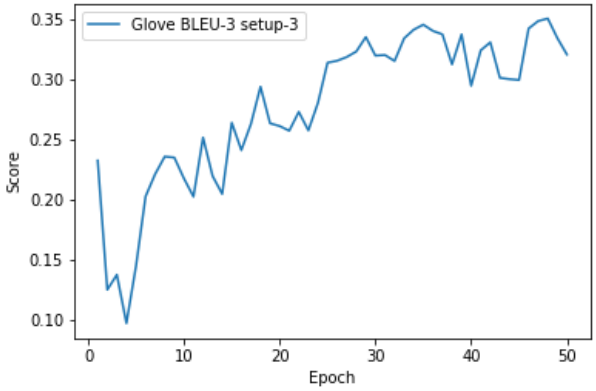
5.6.3 Experiments using Glove

In Table 5.5, we have shown the performance of four different setup while using the Glove model for word vectors. We found that, BLEU-3 and BLEU-4 achieved better score than other metrics for setup-3 and setup-2 respectively. CIDEr and ROUGE also worked better with setup-3 and showed better scores. So, setup-3 acquired better results in almost all the evaluation metrics (BLEU-3, CIDEr, ROUGE).

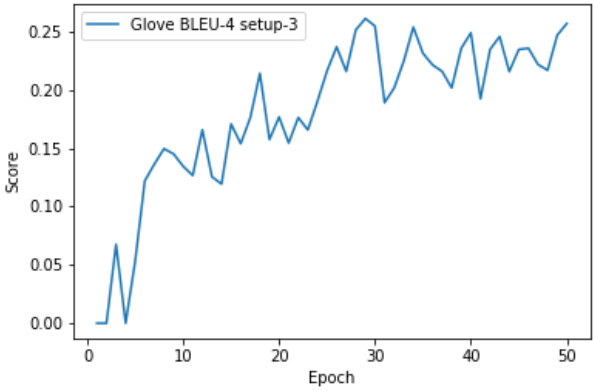
Table 5.5: Performance scores of models with Glove embedding

Setup	BLEU-3	BLEU-4	CIDEr	ROUGE
Setup-1	0.246	0.218	0.253	0.458
Setup-2	0.286	**0.245**	0.124	0.445
Setup-3	**0.314**	0.237	**0.359**	**0.502**
Setup-4	0.273	0.153	0.196	0.424

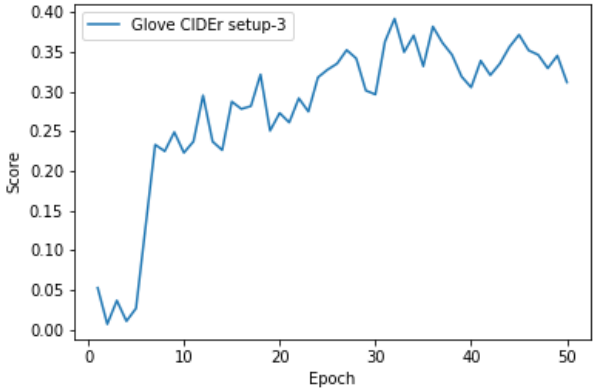
Figure 5.5 shows the validation set results for evaluation matrices over the iterations for setup-3 using Glove embedding model.
In Figure 5.6, a representation of loss curve of the model for setup-3 using Glove embedding on training set is depicted.



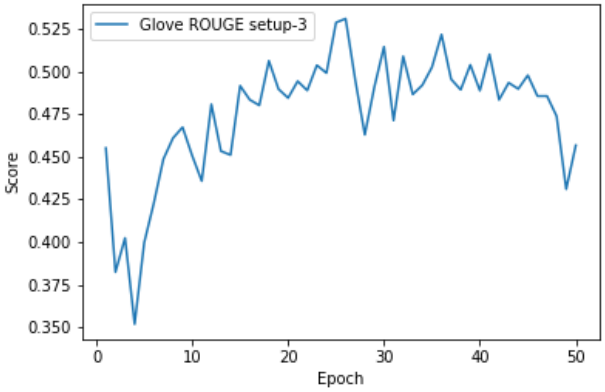
(a) Validation for BLEU-3



(b) Validation for BLEU-4



(c) Validation for CIDEr



(d) Validation for ROUGE

Figure 5.5: Glove Setup-3 Validation Accuracy Curves

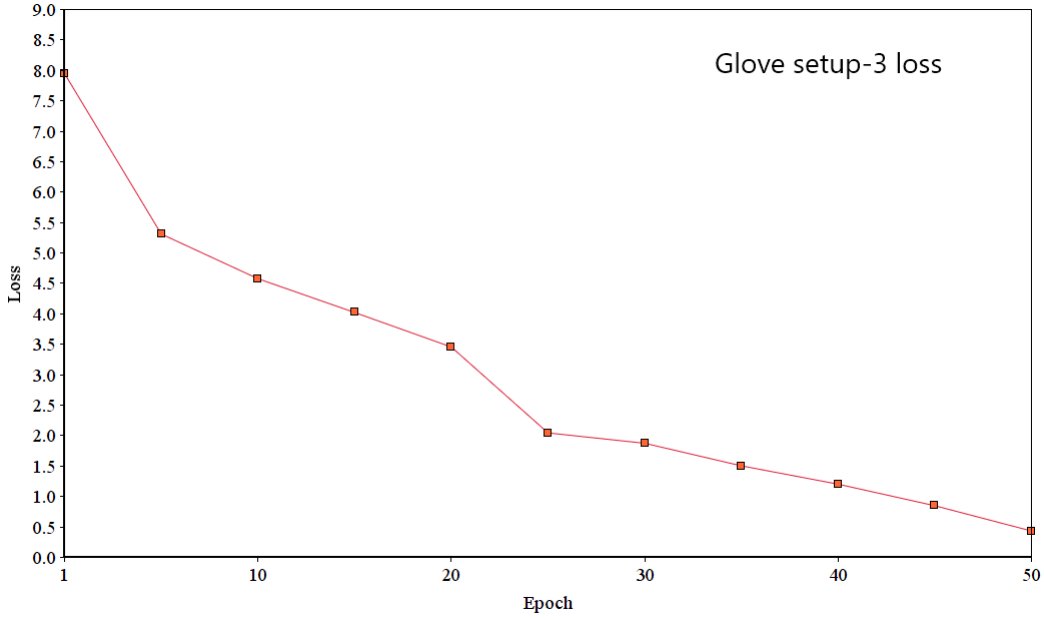


Figure 5.6: Model Loss Curve for Glove embedding with setup-3

5.7 Performance Comparison

We compared the variants of our proposed model on the basis of training time and accuracy. In Figure 5.7, a bar chart of avg. time required per epoch by the models is shown. Setup-1 and Setup-2 required less time because of their smaller hidden sizes in the encoder and decoder. On the other hand, Setup-3 and Setup-4 is relatively large in terms of parameters, as their encoder, decoder hidden sizes are of 1000 and 1500 dimensions respectively as well as their embedding dimension. Setup-4 for the Glove model took almost **3.33h per epoch**, which is the highest among all. Setup-1 of the Word2vec model trained faster with the lowest time required of avg. **1.05h per epoch**.

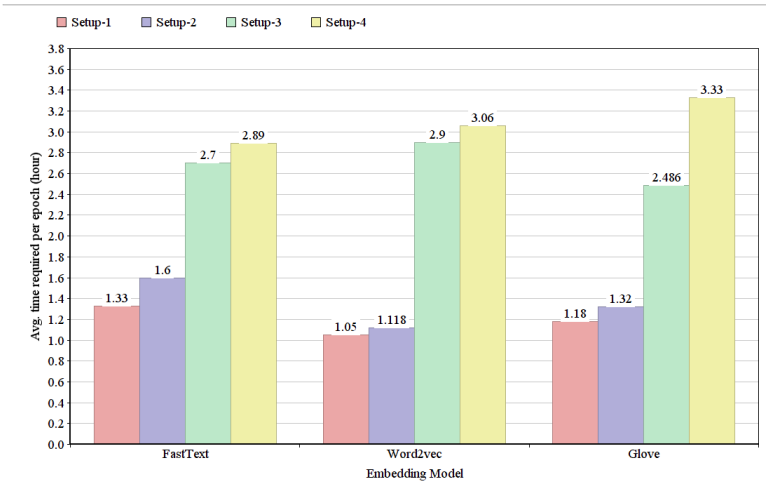


Figure 5.7: Average Training Time for different model setup

Bar chart in Figure 5.8 compares the BLEU-3 scores acquired by the models on test set. The scores varied in a small range of 24% - 32% for most of the models while Setup-3 of Word2vec outperformed with a large margin by getting **43.2%**.

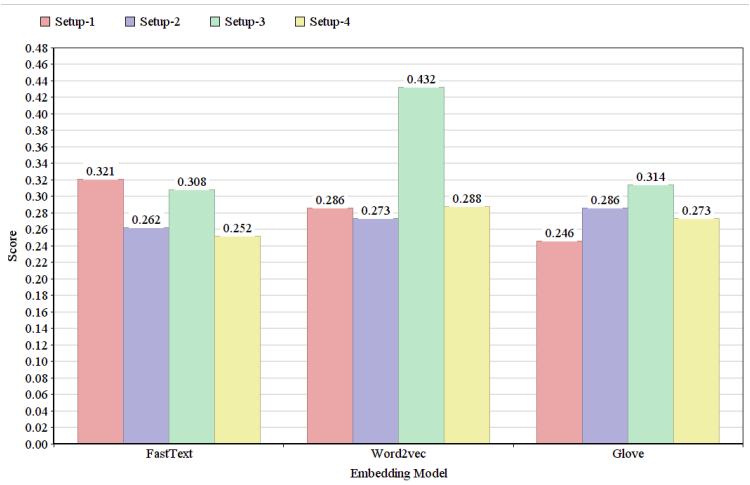


Figure 5.8: Comparison of BLEU-3 scores

Comparison of BLEU-4 is represented in Figure 5.9 which shows that first three setups of FastText model scored nearly 22%, whereas the fourth model struggled with the lowest accuracy of 6%. Almost similar performance was shown by the Glove models with setup-4 being the lowest. In case of Word2vec models, setup-3 retained its top position by scoring the highest **32.6%**.

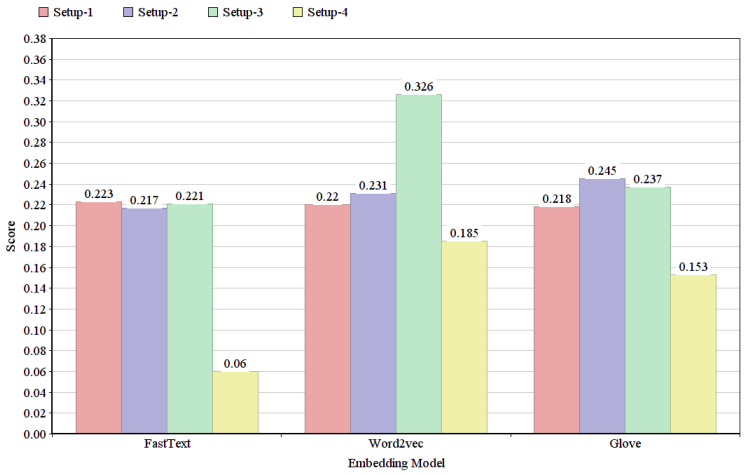


Figure 5.9: Comparison of BLEU-4 scores

Figure 5.10, depicts the comparison for ROUGE scores which are almost uniform with slight deviations. Here also the setup-3 of word2vec model scored the most. It obtained **57.3%**.

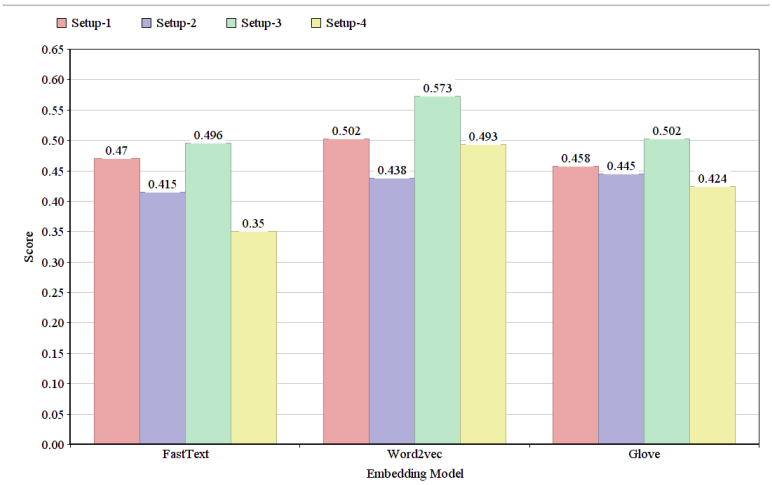


Figure 5.10: Comparison of ROUGE-l scores

CIDEr score comparison is presented in the bar chart of Figure 5.11. Unlike the ROUGE scores, a diversity is observed in case of CIDEr. Setup-1 and setup-3 from all three embedding models performed well in this matrix. Setup-3 of Word2vec scored the highest **51.2%** and topped in all four evaluation matrices.

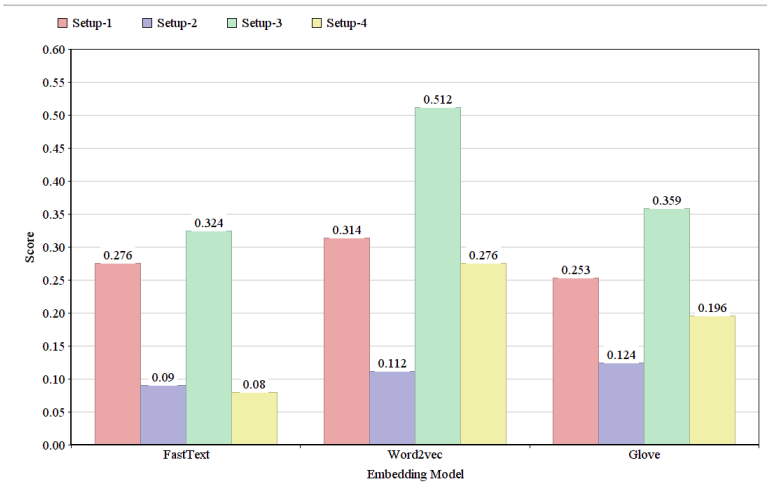


Figure 5.11: Comparison of CIDEr scores

5.8 Performance comparison Between the existing Image Captioning Model and the proposed model

In this section, we will do a comparison among the existing Bengali Image captioning models and our proposed methodology for video captioning as there is no existing work on Bengali Video Captioning. Table 5.6 illustrated the cpmparison between the existing models and our models.

Table 5.6: Performance comparison with the existing Image Captioning models and the proposed model

Paper	Image Dataset	BLEU 3	BLEU 4	CIDER	ROUGEl
	Proposed Model	**43.2**	**32.6**	**51.2**	**57.3**
[5]	BNLIT	32.4	22.8	-	-
[6]	BanglaLekhaImageCaptions	31.7	23.8	-	-
[7]	Flickr8k-BN	33.0	22.0	46.0	54.0

5.9 Qualitative Assessment

In this section, we have presented some pictorial examples (Figure 5.12) of reference and generated sentence of our best performing model, which is Word2vec setup-3 from the test set.

Reference: একটি মহিলা প্যানে তেল যোগ করছেন
Generated: একজন মহিলা একটি পাত্রে জল যোগ করছেন

(a)

Reference:
i. একজন ব্যক্তি একটি মাইক্রোওয়েভে একটি থালা রাখেন এবং এটি শুরু করেন
ii. একজন ব্যক্তি মাইক্রোওয়েভে খাবার রাখছেন
Generated: একজন লোক মাইক্রোওয়েভে একটি কাপ থেকে পানীয় বের করে

(b)

Reference: একজন পুরুষ এবং একজন মহিলা তাদের গাড়িতে পেট্রল রাখছেন
Generated: দুজন লোক একে অপরকে শুভেচ্ছা জানায়

(c)

Reference: কেউ একজন রাইফেল নিয়ে আকাশে একাধিক শিখা গুলি করছে
Generated: একটি গাড়ি সমুদ্রের দিকে উড়ে যাচ্ছে

(d)

Figure 5.12: Examples of Captions Generated by Our Proposed Model

In Figure 5.12, the example 5.12a and 5.12b show that the generated captions are very similar to the reference captions. The context of the images are captured very well. In 5.12c, the generated captions are somehow related, but not similar. The context is not properly captured by the model, so action of the main objects are not described properly. generated captions in the last example, 5.12d are completely irrelevant. So, our proposed model struggles with caption generation when the actions in the video are complex.

Chapter 6

Limitations & Future Works

We tried to contribute to the field of Bengali video captioning which is yet untouched as their is only one research work [46] available on video captioning in Bengali. So, there is no chance of having a look at the previous works on video caption generation in Bengali. Despite of very limited resources, we developed a model that can generate captions from input videos in Bengali leaving us with some success and some limitations. This chapter discusses about the limitations and way of solving the limitations in future works.

6.1 Limitations

There are some limitations of our model for Bengali video captioning.

- The generated captions of the model is has low accuracy. The model can not handle complex visual information in the videos.
- The decoder portion of the model gets short context vector from a single video as input in order to generate the caption which leads to lack of knowledge of the whole context information.
- There is no available dataset with captions in Bengali. So, the captions are translated to Bengali from English using Google Translation API which is not up to mark compared to human translation. This lack of proficiency in language translation leads to improper captioning.
- The model is computationally expensive. In simple words, it takes a long time to train the whole model.

6.2 Future Works

There are some works which can be done in future to upgrade this system to next level.

- Attention mechanism can be used to generate Bengali captions with more accuracy, as attention mechanism can generate captions depending on the whole context.
- Video captioning can not explain the whole perspective of the video clips. So, this model can be upgraded to generate paragraph representing the descriptions of the video. In simple words, rather than generating single sentences, the model should be upgraded to generate multiple sentences.
- A large video dataset can be prepared with multiple Bengali captions for better works in future.
- Including our model, every language model struggles with Bengali complex words and confusing meanings. Resolving all these problems are very important to generate meaningful Bengali captions.

Chapter 7

Conclusion

The field of computer vision is vast, and video captioning works need many improvements. Most of the works regarding video captioning are in the English language. Our works contribute to the task of Bengali video captioning, which has hardly been scratched. There are almost no previous works available, which we had to overcome by gathering knowledge from works regarding video captioning in English. Research works on image captioning in Bengali also help us developing our model. After video and image pre-processing, spatial features are extracted using VGG-19, which is a 2D CNN-based model. Temporal features are extracted using a 3D CNN model, ResNeXt-101. For word embedding, FastText, Glove, and Word2Vec are used. In the encoder and decoder, the sequential model such as Bidirectional Long Short-Term Memory (Bi-LSTM) and two-layer LSTM are used to generate the captions.

We faced problems with the dataset as there is no video dataset available for Bengali video captioning. We solved this problem by translating the available captions in Bengali with Google Translation API in python and encountered another problem regarding the accuracy of the translations as they are not as accurate as human-level translation. We will try to improve the model, as well as prepare a better dataset for further works, which will bring better and precise results.

References

[1] DhakaTribune, "Bangla ranked at 7th among 100 most spoken languages worldwide," "*https://www.dhakatribune.com/world/2020/02/17/bengali-ranked-at-7th-among-100-most-spoken-languages-worldwide*", (accessed: 15.11.2020).

[2] E. Rodriguez, "Bengali language," "*https://www.britannica.com/topic/Bengali-language/additional-info/history*", (accessed: 15.11.2020).

[3] S. Guadarrama, N. Krishnamoorthy, G. Malkarnenkar, S. Venugopalan, R. Mooney, T. Darrell, and K. Saenko, "Youtube2text: Recognizing and describing arbitrary activities using semantic hierarchies and zero-shot recognition," in *Proceedings of the IEEE international conference on computer vision*, pp. 2712–2719, 2013.

[4] S. Cascianelli, G. Costante, T. A. Ciarfuglia, P. Valigi, and M. L. Fravolini, "Full-gru natural language video description for service robotics applications," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 841–848, 2018.

[5] A. Jishan, K. R. Mahmud, A. K. Al Azad, S. Alam, and A. M. Khan, "Hybrid deep neural network for bangla automated image descriptor," *International Journal of Advances in Intelligent Informatics*, vol. 6, no. 2, pp. 109–122, 2020.

[6] A. H. Kamal, M. Jishan, N. Mansoor, *et al.*, "Textmage: The automated bangla caption generator based on deep learning," *arXiv preprint arXiv:2010.08066*, 2020.

[7] T. Deb, M. Z. A. Ali, S. Bhowmik, A. Firoze, S. S. Ahmed, M. A. Tahmeed, N. Rahman, and R. M. Rahman, "Oboyob: A sequential-semantic bengali image captioning engine," *Journal of Intelligent & Fuzzy Systems*, no. Preprint, pp. 1–13, 2019.

[8] Y. Pan, T. Yao, H. Li, and T. Mei, "Video captioning with transferred semantic attributes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6504–6512, 2017.

[9] X. Wang, W. Chen, J. Wu, Y.-F. Wang, and W. Yang Wang, "Video captioning via hierarchical reinforcement learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4213–4222, 2018.

[10] B. Wang, L. Ma, W. Zhang, and W. Liu, "Reconstruction network for video captioning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7622–7631, 2018.

[11] H. Xu, B. Li, V. Ramanishka, L. Sigal, and K. Saenko, "Joint event detection and description in continuous video streams," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 396–405, IEEE, 2019.

[12] J. Zhang and Y. Peng, "Hierarchical vision-language alignment for video captioning," in *International Conference on Multimedia Modeling*, pp. 42–54, Springer, 2019.

[13] S. Ding, S. Qu, Y. Xi, and S. Wan, "A long video caption generation algorithm for big video data retrieval," *Future Generation Computer Systems*, vol. 93, pp. 583–595, 2019.

[14] C. Yan, Y. Tu, X. Wang, Y. Zhang, X. Hao, Y. Zhang, and Q. Dai, "Stat: spatial-temporal attention mechanism for video captioning," *IEEE transactions on multimedia*, vol. 22, no. 1, pp. 229–241, 2019.

[15] J. S. Park, M. Rohrbach, T. Darrell, and A. Rohrbach, "Adversarial inference for multi-sentence video description," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6598–6608, 2019.

[16] M. Rahman, N. Mohammed, N. Mansoor, and S. Momen, "Chittron: An automatic bangla image captioning system," *Procedia Computer Science*, vol. 154, pp. 636–642, 2019.

[17] W. Pei, J. Zhang, X. Wang, L. Ke, X. Shen, and Y.-W. Tai, "Memory-attended recurrent network for video captioning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8347–8356, 2019.

[18] X. Zhang, K. Gao, Y. Zhang, D. Zhang, J. Li, and Q. Tian, "Task-driven dynamic fusion: Reducing ambiguity in video description," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3713–3721, 2017.

[19] M. Rahman, T. Abedin, K. S. Prottoy, A. Moshruba, F. H. Siddiqui, *et al.*, "Semantically sensible video captioning (ssvc)," *arXiv preprint arXiv:2009.07335*, 2020.

[20] Z. Jia and X. Li, "icap: Interactive image captioning with predictive text," in *Proceedings of the 2020 International Conference on Multimedia Retrieval*, pp. 428–435, 2020.

[21] SuperDataScience, "Convolutional neural networks (cnn): Step 1- convolution operation," "*https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-1-convolution-operation*", (accessed: 18.11.2020).

[22] H. Gu, Y. Wang, S. Hong, and G. Gui, "Blind channel identification aided generalized automatic modulation recognition based on deep learning," *IEEE Access*, vol. PP, pp. 1–1, 08 2019.

[23] O. Avci, O. Abdeljaber, S. Kiranyaz, and D. Inman, "Structural damage detection in real time: Implementation of 1d convolutional neural networks for shm applications," in *Structural Health Monitoring & Damage Detection, Volume 7*, pp. 49–54, Springer, 2017.

[24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[25] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[26] Y. Zheng, C. Yang, and A. Merkulov, "Breast cancer screening using convolutional neural network and follow-up digital mammography," p. 4, 05 2018.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[28] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

[29] J. Jordan, "Common architectures in convolutional neural networks," "*https://www.jeremyjordan.me/convnet-architectures*", (accessed: 19.11.2020).

[30] towardsdatascience, "Introduction to recurrent neural network," "*https://towardsdatascience.com/introduction-to-recurrent-neural-network-27202c3945f3*", (accessed: 18.11.2020).

[31] JeremyJordan, "Introduction to recurrent neural networks," "*https://www.jeremyjordan.me/introduction-to-recurrent-neural-networks/*", (accessed: 18.11.2020).

[32] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[33] Colah'sBlog, "Understanding lstm networks," "*http://colah.github.io/posts/2015-08-Understanding-LSTMs/fn1*", (accessed: 18.11.2020).

[34] P. SRIVASTAVA, "Essentials of deep learning : Introduction to long short term memory," "*https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm*", (accessed: 18.11.2020).

[35] TowardsDataScience, "Introduction to word embedding and word2vec," "*https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa*", (accessed: 18.11.2020).

[36] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[37] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[38] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *arXiv preprint arXiv:1607.01759*, 2016.

[39] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

[40] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, *et al.*, "The kinetics human action video dataset," *arXiv preprint arXiv:1705.06950*, 2017.

[41] dumpswikimedia "*https://dumps.wikimedia.org/bnwiki/latest/*", (accessed: 22.09.2020).

[42] D. Chen and W. B. Dolan, "Collecting highly parallel data for paraphrase evaluation," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 190–200, 2011.

[43] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 311–318, Association for Computational Linguistics, 2002.

[44] R. Vedantam, C. Lawrence Zitnick, and D. Parikh, "Cider: Consensus-based image description evaluation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4566–4575, 2015.

[45] C.-Y. Lin, "Rouge: A packagefor automatic evaluation of summaries," in *Proceedingsof Workshop on Text Summarization Branches Out, Post2Conference Workshop of ACL*, 2004.

[46] C. Sur, "Gaussian smoothen semantic features (gssf)–exploring the linguistic aspects of visual captioning in indian languages (bengali) using mscoco framework," *arXiv preprint arXiv:2002.06701*, 2020.

Generated using Undegraduate Thesis LaTeX Template, Version 1.4. Department of Computer Science and Engineering, Ahsanullah University of Science and Technology, Dhaka, Bangladesh.

This thesis was generated on Thursday 26th November, 2020 at 10:06pm.