Hierarchical LSTM with Adjusted Temporal Attention for Video Captioning

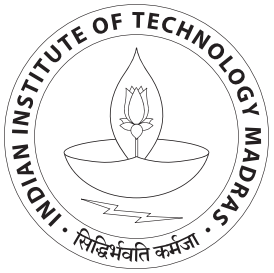
A Project Report

submitted by

MADISETTY UDAY THEJA

in partial fulfilment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY &
MASTER OF TECHNOLOGY



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, MADRAS.
May 2019

THESIS CERTIFICATE

This is to certify that the thesis entitled **Hierarchical LSTM with Adjusted Temporal Attention for Video Captioning**, submitted by **Madisetty Uday Theja** (**CS14B044**), to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelors of Technology** and **Master of Technology**, is a bona fide record of the research work carried out by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. C. Chandra Sekhar
Research Guide
Head of the Department
Dept. of Computer Science and Engineering
IIT-Madras, 600 036

Place: Chennai

Date:

ACKNOWLEDGEMENTS

I would like to thank my guide Prof.C. Chandra Sekhar for his immense support and guidance throughout the tenure of the project. Under his supervision, I was able to explore Machine Learning and Deep Learning and gain valuable insights.

I would also like to thank my faculty advisor Prof. Jayalal Sarma for his guidance throughout my B.Tech and M.Tech.

I also would like to thank my friends and family who stood by me throughout my tenure at the university.

ABSTRACT

KEYWORDS: Video Captioning, Encoder-Decoder Framework, Recurrent Neural Networks, Convolutional Neural Networks, Attention Mechanism

Attention-based neural encoder-decoder frameworks have been widely adopted for video captioning. Most of the existing frameworks force the application of visual attention for every generated word. Words in a sentence can be categorized as visual (eg., ”gun”, ”shooting”) and non-visual (eg., ”a”, ”of”, ”the”). The decoder does not require visual information to predict non-visual words. Also, some words which may seem visual, can be predicted using the context of the sentence. The non-visual words can be predicted without making use of any visual information, by using a natural language model. Forcing attention mechanism on the non-visual words can reduce the efficiency of the model since their corresponding visual signals would not contain any relevant information. This report describes a hierarchical LSTM with adjusted temporal attention (hLSTMat) model which tackles this issue. At every time step, the model determines whether to use the language context information or the visual information. The model also determines which frames to attend to while using the visual signals to capture the relevant information. The report also describes the implementation details and performance studies on the MSVD dataset.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS **i**

ABSTRACT **ii**

LIST OF TABLES **v**

LIST OF FIGURES **vi**

ABBREVIATIONS **vii**

NOTATION **viii**

1 Introduction **1**

1.1 Overview 1

1.2 Organization of Report 2

2 Review of Approaches to Video Captioning **3**

3 Deep Neural Network Models for Sequential Data **5**

3.1 Recurrent Neural Networks 5

3.2 Vanishing and Exploding Gradients 6

3.3 Long Short Term Memory 8

4 Hierarchical LSTM and Attention Models for Video Captioning **10**

4.1 A Basic LSTM for Video Captioning 10

4.2 Hierarchical LSTM with Adjusted Temporal Attention for Video Captioning 11

4.2.1 CNN Encoder 12

4.2.2 Attention Based Hierarchical LSTM Decoder 12

4.3 Temporal Attention Mechanism 15

4.4 Adjusted Temporal Attention Mechanism 16
4.5 Different Types of Decoders 17
4.5.1 Basic LSTM based Decoder 17
4.5.2 Hierarchical LSTM with Temporal Attention (hLSTMt) based Decoder 18
4.5.3 Hierarchical LSTM with Adjusted Temporal Attention (hLSTMat) based Decoder 18
4.6 Mean Aggregation Models 19
4.6.1 Hierarchical LSTM with Adjusted Temporal Attention and Video Feature Means (hLSTMat-PCA) 20
4.6.2 Hierarchical LSTM with Adjusted Temporal Attention and K-Means Clustering (hLSTMat-KMeans) 20

5 Performance Study **22**
5.1 Datasets 22
5.1.1 The Microsoft Video Description Corpus (MSVD) 22
5.2 Implementation Details 23
5.2.1 Preprocessing 23
5.2.2 Training Details 23
5.2.3 Testing Details 24
5.2.4 Evaluation Metrics 24
5.3 The Effect of Different CNN Encoders 25
5.4 Architecture Exploration and Comparison 25
5.4.1 Effect of Attention Mechanism 26
5.4.2 Temporal Attention Analysis 26
5.4.3 Adjusted Temporal Attention Analysis 28
5.4.4 Mean Aggregation Models Analysis 29

6 Conclusion and Future Work **35**
6.1 Conclusion 35
6.2 Future Work 35

LIST OF TABLES

5.1 Dataset Description 22
5.2 Effect of different CNN encoders 25
5.3 Effect of different types of the decoder 26
5.4 Comparison of mean aggregation models 29

LIST OF FIGURES

3.1 Unfolding of a RNN 6

3.2 LSTM Cell 8

4.1 Basic LSTM as a decoder 17

4.2 Hierarchical LSTM with Temporal Attention (hLSTMt) as a decoder 19

4.3 Hierarchical LSTM with Adjusted Temporal Attention (hLSTMat) as a decoder 20

5.1 The ground-truth and model generated captions for three videos in MSVD dataset 27

5.2 Temporal Attention plots for two videos in MSVD dataset using hLSTMat as decoder 31

5.3 Adjusted temporal attention plots for two videos in MSVD dataset using hLSTMat as decoder 32

5.4 Temporal attention values for three frames while generating the word ”man” in the caption 33

5.5 Adjusted temporal attention plots of a video in MSVD dataset comparing hLSTMat, hLSTMat-PCA and hLSTMat-KMeans as decoder 34

ABBREVIATIONS

MLP	Multi Layer Perceptron
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
BPTT	Back Propagation Through Time

NOTATION

$\varnothing$	Hyperbolic tangent function, *tanh*
σ	Logistic sigmoid function
$\odot$	Element wise multiplication
Θ	Model parameters
$\mathscr{L}(\Theta)$	Loss function of model with parameters Θ

CHAPTER 1

Introduction

1.1 Overview

Video captioning is the task of automatically annotating videos with natural language descriptions. It has several applications such as automatic video subtitling, describing movies for the visually impaired, improving video indexing and search quality of online videos. Videos contain diverse set of objects, scenes, attributes, salient content and complex interactions between the actors which amount to a vast amount of information. Handling such a huge amount of information present in videos is a challenge. Moreover, the generated captions must follow natural language semantics. Image captioning takes a single image as the input and produces a caption of variable length. However, different videos have different number of frames. Thus, video captioning must additionally handle a variable length input. This makes the task of video captioning challenging.

The development of Long Short Term Memory (LSTM) units and deep Convolutional Neural Networks (CNN) has led to advances in the fields of image captioning, speech recognition and language translation. Inspired by these advances there have been attempts for video captioning.

Early attempts on video captioning were based on the encoder-decoder framework where a visual convolution model was directly connected to a LSTM network. The idea was that the CNN encoder would encode the input video into a compact

representation with all the relevant visual features and the LSTM decoder would then decode the visual information into a natural language description. Later, incorporating attention mechanism improved the performance of these models. However, the visual attention models attend to the video frames at each time step without considering any language context information of the predicted word.

Words in a sentence can be categorized as visual (eg., ”gun”, ”shooting”) and non-visual (eg., ”a”, ”of”, ”the”). Also, some words which may seem visual, can be predicted using the context of the sentence. The non-visual words can be predicted without making use of any visual information, by using a natural language model. Forcing attention mechanism on the non-visual words can reduce the efficiency of the model since their corresponding visual signals would not contain any relevant information.

This report describes a unified encoder-decoder framework named hLSTMat, a hierarchical LSTM with adjusted temporal attention model for video captioning [1] which tackles these issues.

1.2 Organization of Report

This report is organized as follows. Chapter 2 presents the literature survey of the various approaches to video captioning. Chapter 3 presents the overview of required basic blocks such as RNN and LSTM. Chapter 4 presents the hierarchical LSTM with adjusted temporal attention model and its variants for video captioning. Chapter 5 gives the details of implementation and experimental results. Chapter 6 presents the conclusion and the directions for future work.

CHAPTER 2

Review of Approaches to Video Captioning

An encoder-decoder neural network framework was used by Venugopalan et al.[2] for video captioning. The model directly concatenates a deep neural network with a recurrent neural network. A previously trained convolutional neural network [3] was used to capture the visual features for each frame of the video. To deal with the varying length of the video, a single vector was computed by taking the mean of the feature vectors of each frame to represent the entire video. However, averaging all the frames irrespective of their temporal relationships into a single vector, ignores most of the temporal information. This may result in loss of information. For example, we cannot differentiate the order in which the different objects appear from the averaged features.

Based on the soft alignment method[4], Yao et al.[5] used a temporal attention mechanism to utilize the temporal structure of the video. The temporal structure refers to the order in which the objects, actions and scenes appear in the video. This approach tackles the shortcomings of [2] in the following ways. Firstly, a 3-D convolutional neural network pre-trained for action recognition is used to get the spatio temporal motion features. For feature extraction, the videos are assumed to have fixed volume (width, height, time) i.e same dimensions for each frame and same number of frames for each video are chosen. Dense trajectory features (HoG, HoF, MBH) are extracted and concatenated. Secondly, an attention mechanism is incorporated which attends to a relevant subset of frames which the decoder can use to describe only the actions or objects in that set of frames by weighting the

frame features non-uniformly.

The visual attention models attend to the video frames at different time steps without considering any language context information of the predicted word. For generating visual words (eg., ”gun”, ”shooting”) which contain the corresponding visual signals, attending to the visual features is useful. For non-visual words (eg., ”a”, ”of”, ”the”) there is no corresponding visual signal, so attending to the visual features is misleading and would reduce the performance of the system. To tackle this issue, Lu et al.[6] introduced an adaptive attention encoder-decoder framework for the task of image captioning, which can automatically determine when to use the language context information and when to use visual information for predicting the next word.

Song et al.[1] extended the adaptive attention mechanism for image captioning [6] to the domain of video captioning and used a unified encoder-decoder framework named hLSTMat, a hierarchical LSTM with adjusted temporal attention model for video captioning.

CHAPTER 3

Deep Neural Network Models for Sequential Data

3.1 Recurrent Neural Networks

Recurrent Neural Networks (RNN) are used to model sequence data. The main characteristics of sequence data are:

- Successive inputs are dependent on each other
- The input and output sizes are not fixed
- Same function is performed on the input at every time step

Consider the task of predicting the next word in a sentence. To perform this task, we need to know the previous words. Hence, the successive inputs are dependent on each other. Also, the number of words varies for each sentence. Hence, the input sizes are variable.

The RNNs are called *recurrent* because they are multiple copies of the same network performing the same task on an input element at every time step with the output depending on the information captured from all the previous computations. The state of the RNN can be viewed as the *memory* unit of RNN which contains useful information about what has been calculated until the current time step. In theory, we expect the RNN to capture information in sequences of arbitrary length. But in practice, they are limited to capturing information up to only a few steps prior to the current time step.

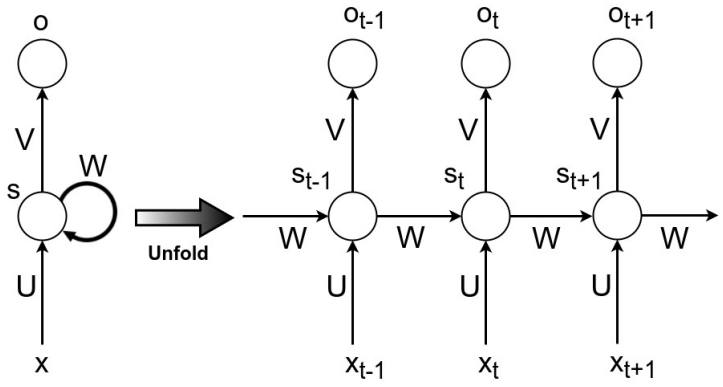


Figure 3.1: Unfolding of a RNN

Figure 3.1 shows the unrolling of a RNN in a forward propagation. The number of times the RNN is unrolled depends on the required number of time steps. For example, if we have a sentence with 3 words, then we unroll the RNN 3 times. The equations for the computations in the RNN are shown in Equations 3.1 and 3.2.

$$s_t = \sigma(Ux_t + Ws_{t-1} + b) \tag{3.1}$$

$$o_t = softmax(Vs_t + c) \tag{3.2}$$

Here, x_t is the input , s_t is the state of RNN and o_t is the output at each time step t. The parameters of the network are $W \in \mathbb{R}^{D\times D}$, $U \in \mathbb{R}^{D\times N}$ and $V \in \mathbb{R}^{K\times D}$ where $x_t \in \mathbb{R}^N$, $s_t \in \mathbb{R}^D$ and $o_t \in \mathbb{R}^K$ at each time step t. N is the dimension of the input x_t, D is the dimension of the RNN's state s_t and K is the dimension of the output o_t. The RNN shares the same parameters U, V and W across all time steps.

3.2 Vanishing and Exploding Gradients

The RNNs are trained using back propagation through time (BPTT) by computing gradients with respect to each of the parameters. The loss function $\mathscr{L}(\theta)$ where

θ corresponds to all the parameters of the model is the sum of losses at each time step $\mathscr{L}_t(\theta)$, given by equation 3.3 .

$$\mathscr{L}(\theta) = \sum_{t=1}^{T} \mathscr{L}_t(\theta) \tag{3.3}$$

For back propagation, we need to compute the gradients with respect to W, U and V. Consider the derivative with respect to the weight matrix W.

$$\frac{\partial \mathscr{L}(\theta)}{\partial W} = \sum_{t=1}^{T} \frac{\partial \mathscr{L}_t(\theta)}{\partial W} \tag{3.4}$$

Using the chain rule of derivatives, $\frac{\partial \mathscr{L}_t(\theta)}{\partial W}$ is the sum of gradients along all paths from $\mathscr{L}(\theta)$ to W.

$$\frac{\partial \mathscr{L}_t(\theta)}{\partial W} = \sum_{i=1}^{T} \frac{\partial \mathscr{L}_t(\theta)}{\partial s_i} \frac{\partial s_i(\theta)}{\partial W} \tag{3.5}$$

From Equation 3.1, s_i depends on s_{i-1}. So we compute the explicit gradient $\frac{\partial^+ s_i(\theta)}{\partial W}$ by treating all other inputs as constant and the implicit gradient by summing over all paths from s_i to W.

$$\frac{\partial s_t(\theta)}{\partial W} = \sum_{i=1}^{T} \frac{\partial s_t(\theta)}{\partial s_i} \frac{\partial^+ s_i(\theta)}{\partial W} \tag{3.6}$$

$$\frac{\partial s_t(\theta)}{\partial s_i} = \frac{\partial s_t(\theta)}{\partial s_{t-1}} \frac{\partial s_{t-1}(\theta)}{\partial s_{t-2}} \cdots \frac{\partial s_{i+1}(\theta)}{\partial s_i} \tag{3.7}$$

$$\frac{\partial s_t(\theta)}{\partial s_i} = \prod_{j=i}^{t-1} \frac{\partial s_{j+1}(\theta)}{\partial s_j} \tag{3.8}$$

From Equation 3.8 we can observe that the product can explode or vanish if the values of $\frac{\partial s_{j+1}(\theta)}{\partial s_j}$ are greater than or less than 1 respectively, if the number of time steps t is large. This makes training of RNNs difficult. Thus, we cannot capture long range dependencies using standard RNN architecture.

3.3 Long Short Term Memory

Long Short Term Memory networks (LSTMs) are a variant of RNNs which can learn long range dependencies. LSTMs tackle the problem of vanishing and exploding gradients using three gates: input gate i_t, forget gate f_t and output gate o_t.

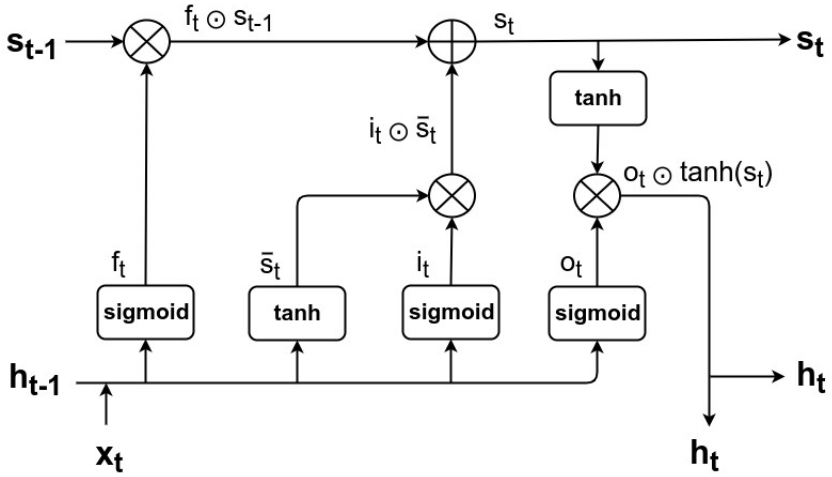


Figure 3.2: LSTM Cell

The Figure 3.2 shows the structure of a LSTM cell. The main components of a LSTM cell are:

- **Cell State** (c_t): The current state or memory cell of the network.
- **Hidden State** (h_t): The output of the LSTM cell.
- **Input Gate** (i_t): Controls the state by selectively reading the input.
- **Forget Gate** (f_t): Controls the state by selectively forgetting the previous state.

- **Output Gate** (o_t): Controls the hidden state by selectively writing from the current state.

The equations for the computations in a LSTM cell are given by :

$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o) \tag{3.9}$$

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i) \tag{3.10}$$

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f) \tag{3.11}$$

$$\tilde{s_t} = \varnothing(W h_{t-1} + U x_t + b) \tag{3.12}$$

$$s_t = f_t \odot s_{t-1} + i_t \odot \tilde{s_t} \tag{3.13}$$

$$h_t = o_t \odot \varnothing(s_t) \tag{3.14}$$

where W_o, W_i, W_f, U_o, U_i, U_f, b_o, b_i, b_f, b, U and W are parameters to be learned. x_t is the input vector to the LSTM at each time step t. $\varnothing$ represents the hyperbolic tangent function tanh. σ represents the logistic sigmoid function. $\odot$ represents element wise multiplication.

CHAPTER 4

Hierarchical LSTM and Attention Models for Video Captioning

Video captioning is a sequence learning problem. The encoder-decoder framework with a deep CNN network as an encoder and LSTM units as a decoder is used. A sequence of video frames is the input to the model and a sequence of words, which is of variable length is the output. The organization of this section is as follows: Section 4.1 describes how a basic LSTM is used as a decoder for the video captioning task and Section 4.2 describes a hierarchical LSTM with adjusted temporal attention model for video captioning which is an encoder-decoder framework named hLSTMat.

4.1 A Basic LSTM for Video Captioning

As described in Chapter 3, RNNs are used to model the sequential data. LSTMs are a variant of RNNs which tackle the problem of vanishing and exploding gradients. For the sake of convenience, let us denote the function updating the internal state of a LSTM unit as:

$$h_t, s_t = LSTM(x_t, h_{t-1}, s_{t-1}) \tag{4.1}$$

where h_t is the hidden state, s_t is the cell state and x_t is the input at each time step t.

A deep convolutional neural network is used to encode every frame of the given video input. Let v denote the video input and $\varnothing_E$ denote the CNN neural network.

$$V = \{v_1, v_2, \ldots, v_n\} = \varnothing_E(v) \tag{4.2}$$

where n denotes the number of frames in the video v, $v_i \in \mathbb{R}^d$ denotes feature representation of i-th frame, with dimension d. An LSTM is used as a decoder which models V to generate a caption $y = \{y_1, y_2, \ldots, y_T\}$ for the given video v where T is number of words in the caption.

V is used to compute the initial cell state s_0 and hidden state h_0 of the LSTM and at each time step t, LSTM makes use of the previous hidden state h_{t-1}, previous output y_{t-1} and video representation V to update its current internal hidden state h_t and emit the word y_t.

$$h_t, y_t = \varnothing_D(h_{t-1}, y_{t-1}, V) \tag{4.3}$$

The decoder $\varnothing_D$ is run sequentially over the output sequence until y_t = <EOS>, the end-of-sequence tag is generated.

4.2 Hierarchical LSTM with Adjusted Temporal Attention for Video Captioning

The hLSTMat model has two components: 1) CNN Encoder and 2) attention based hierarchical LSTM decoder.

4.2.1 CNN Encoder

Deep CNN networks can compute compact and representative features which can capture relevant visual information. The fully connected or intermediate convolutional layers of a CNN are used as visual features. In case of videos, image trained CNNs are used to get features of each frame of the video.

4.2.2 Attention Based Hierarchical LSTM Decoder

The decoder consists of two LSTMs to simultaneously consider both visual and language context information. The visual information is decoded by the bottom LSTM layer and the language context information is mined by the top LSTM layer. The decoder also has two attention mechanisms. Temporal attention is used to select specific frames and adjusted temporal attention is used to determine whether to use language context information or visual information. The multilayer perceptron layer is used to compute the probabilities of all words in the vocabulary for predicting the next word in the caption.

Bottom LSTM Layer

A sentence is treated as a sequence of words. At every time step, the word embedding feature vector for the word x_t is given as the input to the LSTM. The continuous representation space of the video V is used to compute the initial cell state s_0 and hidden state h_0 of the bottom LSTM layer and the LSTM makes use of the previous hidden state h_{t-1}, previous cell state s_{t-1} and current word x_t to update

its internal hidden state h_t and cell state s_t for each time step t.

$$h_0, s_0 = [W^{ih}; W^{ic}]Mean(\{v_i\}) \tag{4.4}$$

$$h_t, s_t = LSTM(x_t, h_{t-1}, s_{t-1}) \tag{4.5}$$

where $x_t = E[y_t]$ denotes the word embedding of a single word x_t. $Mean(.)$ denotes the average of all the features in v_i. W^{ih} and W^{ic} are parameters to be learned.

Top LSTM Layer

The output of the bottom LSTM layer is fed as the input to the top LSTM layer. The continuous representation space of the video V is used to compute the initial cell state $\bar{s}_0$ and hidden state $\bar{h}_0$ of the top LSTM layer and the LSTM makes use of the previous hidden state $\bar{h}_{t-1}$, previous cell state $\bar{s}_{t-1}$ and output of bottom LSTM unit h_t to update its internal hidden state $\bar{h}_t$ and cell state $\bar{s}_t$ for each time step t.

$$\bar{h}_t, \bar{s}_t = LSTM(h_t, \bar{h}_{t-1}, \bar{s}_{t-1}) \tag{4.6}$$

Attention Layers

Attention layers are used to efficiently choose visual information or language context information and also select which frames to attend to. The temporal attention model computes the context vector c_t at every time step t using the output of the bottom LSTM layer h_t and the video features V.

$$c_t = \varphi(h_t, V) \tag{4.7}$$

where φ denotes the function of temporal attention model. The final context vector through the adjusted gate in adjusted temporal mechanism is denoted by $\bar{c}_t$.

$$\bar{c}_t = \psi(h_t, \bar{h}_t, c_t) \tag{4.8}$$

where ψ denotes the adjust gate function. The temporal attention mechanism and adjusted temporal attention are described in detail in Sections 4.3 and 4.4.

MLP Layer

The output of the bottom LSTM layer h_t and the final context vector through the adjusted gate $\bar{c}_t$ are used to generate the probabilities of all the words in the vocabulary.

$$p_t = softmax(U_p\phi(W_p[h_t; \bar{c}_t] + b_p) + d) \tag{4.9}$$

where U_p, W_p, b_p and d are parameters to be learned. At each time step t the softmax layer output p_t is interpreted as the probability distribution over the possible words:

$$P(x_t|x_{<t}, V, \Theta) \tag{4.10}$$

where Θ denotes the model parameters and V denotes the feature set of the input video. Cross entropy function is used to compute the cost. The total cost is the summation of cost at each time step t. Hence, to learn the parameters of the model Θ, we minimize the negative logarithm of the likelihood:

$$min_\Theta - \sum_{t=1}^{T} \log P(x_t|x_{<t}, V, \Theta) \tag{4.11}$$

where T is the number of time steps. This is the objective function to optimize the model.

4.3 Temporal Attention Mechanism

Temporal attention is used to compute the context vector c_t. Videos have variable length. To compute a fixed size representation of the video, a simple method would be to find the average of features across the video.

$$c_t = \frac{1}{n}\sum_{i=1}^{n} v_i \tag{4.12}$$

Averaging all the frames irrespective of their temporal relationships into a single vector, ignores most of the temporal information. This may result in losing information. Instead of using this simple averaging strategy, dynamic weighted sums of the features are calculated. The feature representation of the video V and the bottom LSTM layer's hidden state h_t are fed through a single layer neural network. Then, $softmax$ function is used to compute the attention weights over n frames, at each time step t.

$$\epsilon_t = w^T \tanh(W_a h_t + U_a V + b_a) \tag{4.13}$$

$$\alpha_t = softmax(\epsilon_t) \tag{4.14}$$

where w^T, W_a, U_a and b_a are the parameters to be learned. $\alpha_t \in \mathbb{R}^n$ gives the relevance of each frame at a time step t. The attention weights α_t^i learned through soft attention using Equation 4.14 are used compute the context vector c_t by taking

the dynamic weighted sum of the video features at each time step t:

$$c_t = \frac{1}{n} \sum_{i=1}^{n} \alpha_t^i v_i \tag{4.15}$$

where the number of frames in the video is denoted by n.

4.4 Adjusted Temporal Attention Mechanism

Adjusted temporal attention is used to generate the final context vector $\bar{c}_t$ in Equation 4.8, to ensure that the model uses the relevant visual information to predict visual words and barely uses any visual information to predict non-visual words. Forcing attention mechanism on the non-visual words can reduce the efficiency of the model since their corresponding visual signals would not contain any relevant information. The hidden state of the bottom LSTM layer h_t stores both visual and linguistic information. h_t is the latent representation of what is already known by the decoder. h_t is used to compute the adjusted gate β_t:

$$\beta_t = sigmoid(W_s h_t) \tag{4.16}$$

$$\bar{c}_t = \beta_t c_t + (1 - \beta_t)\bar{h}_t \tag{4.17}$$

where W_s is the parameter to be learned. β_t is the adjusted gate which is projected to the range $[0, 1]$. When $\beta_t = 1$ then complete visual information is considered using the context vector c_t, while when $\beta_t = 0$ then no visual information is considered and the next word is predicted using only the language context model.

4.5 Different Types of Decoders

4.5.1 Basic LSTM based Decoder

When a single layer basic LSTM, described in Section 4.1, is used as a decoder without any attention mechanisms, the output of the MLP layer which represents the probability distribution over the possible words is now represented as:

$$p_t = softmax(U_p\phi(W_p[h_t] + b_p) + d) \tag{4.18}$$

where h_t is the output of the LSTM and U_p, W_p, b_p, d are parameters to be learned.

Figure 4.1 illustrates the architecture of the model which uses a single layer basic LSTM as a decoder without any attention mechanisms, where x_t is the word embedding input to the decoder, h_t is the hidden state of the LSTM and y_t is the output of MLP layer, at each time step t.

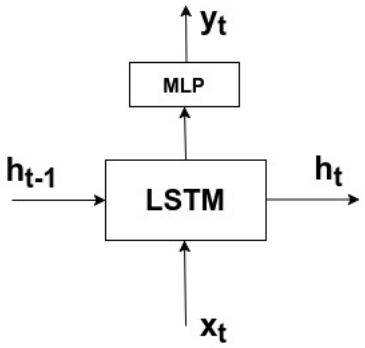


Figure 4.1: Basic LSTM as a decoder

4.5.2 Hierarchical LSTM with Temporal Attention (hLSTMt) based Decoder

When a hierarchical LSTM with only temporal attention (described in Section 4.3) is used as a decoder without any adjusted temporal attention mechanism (described in Section 4.4), the final context vector and the output of the MLP layer which represents the probability distribution over the possible words are now represented as:

$$\bar{c}_t = c_t + \bar{h}_t \tag{4.19a}$$

$$p_t = softmax(U_p\phi(W_p[h_t; \bar{c}_t] + b_p) + d) \tag{4.19b}$$

where h_t is the output of the bottom LSTM, $\bar{h}_t$ is the output of the top LSTM layer, c_t is the context vector, $\bar{c}_t$ is the final context vector and U_p, W_p, b_p, d are parameters to be learned..

Figure 4.2 illustrates the architecture of the hLSTMt model which uses a hierarchical LSTM with only temporal attention, where x_t is the word embedding input to the decoder, h_t is the hidden state of the bottom LSTM, $\bar{h}_t$ is the hidden state of the top LSTM, V is the feature representation of video, c_t is the context vector and y_t is the output of MLP layer, at each time step t.

4.5.3 Hierarchical LSTM with Adjusted Temporal Attention (hLSTMat) based Decoder

When a hierarchical LSTM with both temporal attention and adjusted temporal attention mechanism described in Section 4.2, is used as a decoder, we incorporate both temporal attention and adjusted temporal attention.

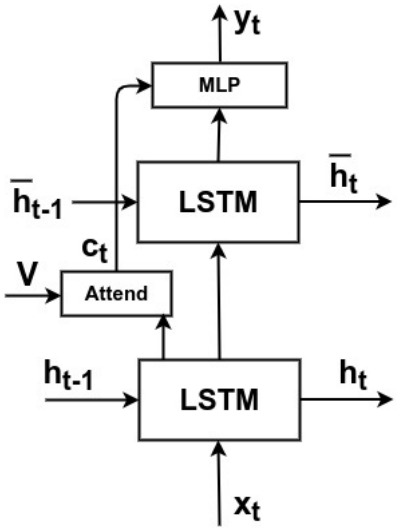


Figure 4.2: Hierarchical LSTM with Temporal Attention (hLSTMt) as a decoder

Figure 4.3 shows the architecture of the hLSTMat model which uses a hierarchical LSTM with both temporal attention and adjusted temporal attention, where x_t is the word embedding input to the decoder, h_t is the hidden state of the bottom LSTM, $\bar{h}_t$ is the hidden state of the top LSTM, V is the feature representation of video, c_t is the context vector, $\bar{c}_t$ is the final context vector and y_t is the output of MLP layer, at each time step t.

4.6 Mean Aggregation Models

When a hierarchical LSTM with both temporal attention and adjusted temporal attention mechanism described in Section 4.5.3, is used as a decoder, the temporal attention weights for each video frame, given by Equation 4.13 and the adjusted temporal attention weights, given by Equation 4.16 were not being learned properly, possibly due to the large number of parameters. In this section, few mean aggregation models are explored which try to reduce the number of parameters and learn the attention weights better.

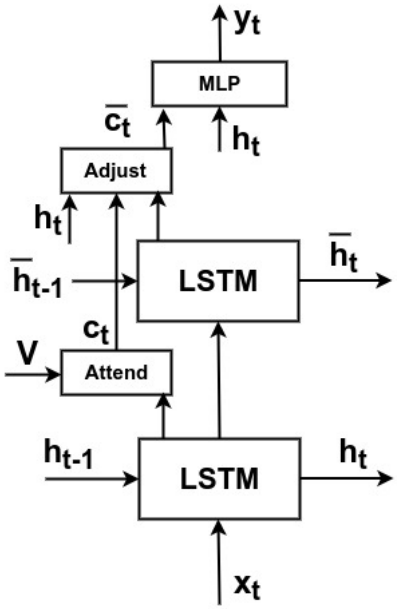


Figure 4.3: Hierarchical LSTM with Adjusted Temporal Attention (hLSTMat) as a decoder

4.6.1 Hierarchical LSTM with Adjusted Temporal Attention and Video Feature Means (hLSTMat-PCA)

To reduce the dimensionality of the video feature representation, Principal component analysis (PCA) is used. The dimension of the mean of the video features of each video is reduced to the LSTM size using PCA. This is used as the new video representation V in Equation 4.13.

4.6.2 Hierarchical LSTM with Adjusted Temporal Attention and K-Means Clustering (hLSTMat-KMeans)

To reduce the dimensionality of the video feature representation, the output of the CNN is clustered using K-Means algorithm with 3 k-centers.

$$\bar{e}_t = w_a^T \tanh(W_a h_t + U_a \bar{V} + b_a) \tag{4.20}$$

$$\epsilon_t = w^T \bar{e}_t + b \tag{4.21}$$

$$\alpha_t = softmax(\epsilon_t) \tag{4.22}$$

$$c_t = \frac{1}{n} \sum_{i=1}^{n} \alpha_t^i v_i \tag{4.23}$$

where w^T, w_a^T, W_a, U_a, b and b_a are the parameters to be learned and the number of frames in the video is denoted by n. The video $\bar{V}$ in Equation 4.20, is now represented as *(k-centers, mean at each k-center)*. The weights for the k-centers are then passed through a single layer neural network to get the weights for each frame of the video given by Equation 4.21. $\alpha_t \in \mathbb{R}^n$ gives the relevance of each frame at a time step t, v_i is the CNN output of the i^{th} video frame and c_t is the context vector.

CHAPTER 5

Performance Study

This section describes the implementation details of the hierarchical LSTM with adjusted temporal attention (hLSTMat) discussed in Section 4.2 and its variants. First, the effect of different CNN encoders is studied. Then the influence of the different components of the network is explored.

5.1 Datasets

The widely used publicly available datasets for the task of video captioning are considered.

5.1.1 The Microsoft Video Description Corpus (MSVD)

This dataset contains 1970 short videos with various natural language annotations for every video. It has approximately 80000 human annotated video-description pairs. The dataset is split into validation, test and training set with 100, 670 and 1200 videos respectively.

Table 5.1: Dataset Description

Name	#Total	#Training	#Validation	#Testing	#Vocabulary
MSVD	1970	1200	100	670	9433

5.2 Implementation Details

5.2.1 Preprocessing

For MSVD dataset, only the captions described in English are filtered. All the captions are then converted to lower case, punctuations are removed and then tokenized. The dataset is split into validation, test and training set. The vocabulary is built over the training split which yields a set of 9430 words. The begin of sentence <BOS>, end of sentence <EOS> and unknown word UNK are added to the vocabulary. Each word in the vocabulary is given a unique integer identifier and a reverse vocabulary is also generated which maps the unique identifiers to the words in the vocabulary.

Sentences have variable lengths. To deal with this issue, for each sentence a begin of sentence <BOS> tag is added at the start of the sentence and an end of sentence tag <EOS> is added to the end of the sentence. Using the vocabularies, each caption is then converted into a sequence of integer identifiers.

Every video is preprocessed by dividing the first 360 frames into 28 chunks and selecting the first frame from each chunk. Each frame is then fed through a CNN network and the visual features are extracted. Thus, for a ResNet-50 CNN encoder [7], for every frame we extract the output of the pool5 layer which is a 2,048 dimensional feature vector.

5.2.2 Training Details

The word embedding size is set as 512 and the LSTM size is set as 512. In the training phase, each word is input at the corresponding time step and using a

mini-batch size of 64 for the training video-sentence pairs,the objective function Equation 4.11 is optimized. The adadelta optimizer [8] is used to optimize the loss function. To prevent over fitting, dropout regularization of 0.5 is used. The model is trained over 500 epochs with a patience set to 20.

5.2.3 Testing Details

For testing the model, the input is the <BOS> tag and the output at a time step is fed as the input for the next time step. This process continues until <EOS> is generated. For each word generation, the output can either be chosen by beam search or stochastically by choosing the word with the highest probability.

Beam search [9] is a searching strategy which chooses the best k sentences at each timestep t as candidates $\bar{S}$ to generate the sentences at the next timestep $t + 1$ and keeps only best k results of them iteratively. The beam size is denoted by k. Finally, we choose the best sentence from the final set of candidates by approximating $S = argmax_{\bar{S}} Pr(\bar{S}|X)$. In the entire experiment, beam size of 5 is used.

5.2.4 Evaluation Metrics

The generated captions are evaluated based on two standard data evaluation metrics: BLEU [10] and METEOR [11].

5.3 The Effect of Different CNN Encoders

This experiment evaluates the hierarchical LSTM with adjusted temporal attention model (hLSTMat) using different CNN encoders to study their influence. The different CNN encoders used are VGG19 [12], ResNet-50, ResNet-152 [7] and Inception-v3 [13]. The dataset used was MSVD dataset and hLSTMat described in Section 4.2 was used as the decoder. The results are shown in Table 5.2.

Table 5.2: Effect of different CNN encoders

Model	BLEU@1	BLEU@2	BLEU@3	BLEU@4	METEOR
VGG19	81.2	70.1	61.0	51.1	31.7
ResNet-50	83.1	73.0	64.2	54.6	32.9
ResNet-152	83.0	73.5	65.1	55.3	33.7
Inception-v3	83.8	74.0	65.3	55.7	33.6

It is observed that Inception-v3 performs the best with BLEU@4 score of 55.7% and METEOR score of 33.6%. ResNet-152 is very close in performance to Inception-v3 with BLEU@4 score of 55.3% and METEOR score of 33.7%. ResNet-152 seems to perform better than ResNet-50.

5.4 Architecture Exploration and Comparison

This experiment explores the architecture and studies the influence of the attention mechanisms. Three different variants of the proposed model are compared. A basic LSTM described in Section 4.5.1, hierarchical LSTM with temporal attention but no adjusted temporal attention (hLSTMt) 4.5.2, and hierarchical LSTM with adjusted temporal attention (hLSTMat) described in Section 4.5.3, as a decoder. The experiments are conducted on MSVD dataset and use Inception-v3 as the

CNN encoder. The results are shown in Table 5.3.

Table 5.3: Effect of different types of the decoder

Decoder	BLEU@1	BLEU@2	BLEU@3	BLEU@4	METEOR
Basic LSTM	79.6	69.7	60.3	49.8	32.1
hLSTMt	83.0	72.8	63.2	53.6	32.9
hLSTMat	83.8	74.0	65.3	55.7	33.6

It is observed that hLSTMt model performs better than Basic LSTM model and hLSTMat model performs better than hLSTMt model. This shows that attention mechanism is improving the performance of the model.

5.4.1 Effect of Attention Mechanism

Figures 5.1a, 5.1b and 5.1c show the actual and generated captions by basic LSTM model, hLSTMt model and hLSTMat model for three videos from MSVD dataset. In Figure 5.1a we observe that the hLSTMt model is able to detect that the man is running by exploiting the temporal structure and the hLSTMat model is further able to detect that it is a soccer player and generates a caption much closer to the ground-truth. Similarly in Figure 5.1b, hLSTMt model detects that the animal is "walking" and hLSTMat model further uses language context information and generates "walking through a field". Similarly in figure 5.1c, hLSTMat model detects that it is "eye makeup".

5.4.2 Temporal Attention Analysis

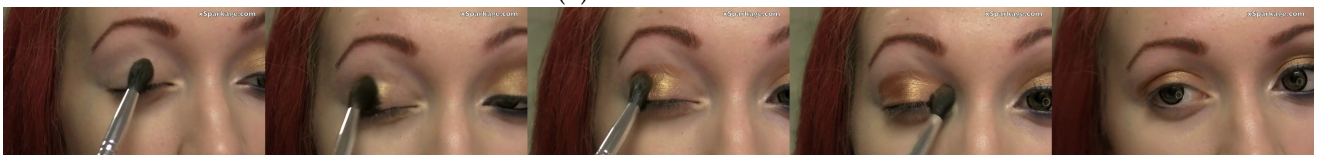
Figures 5.2a and 5.2b illustrate the temporal attention weights given by Equation 4.14 for a frame while generating a particular word in the caption.

Truth : a soccer player making a long goal
Basic LSTM sample : two teams are playing
hLSTMt sample : a man is running
hLSTMat sample : a soccer player makes a goal

(a)

Truth : a jackal is walking around in a field
Basic LSTM sample : a dog is catching a fish
hLSTMt sample : a dog is walking
hLSTMat sample : a animal is walking through a field

(b)



Truth : a woman is putting on gold eyeshadow
Basic LSTM sample : a woman is putting a stick in her mouth
hLSTMt sample : a girl is applying makeup
hLSTMat sample : a woman is applying eye makeup

(c)

Figure 5.1: The ground-truth and model generated captions for three videos in MSVD dataset

In Figure 5.2a the caption is ”a man is shooting a gun”. It is observed that while predicting the word ”man”, higher weightage was given to the first frame. While predicting the word ”shooting”, higher weightage was given to the second frame which depicts the shooting and recoil action of the gun. While predicting the word ”gun”, higher weightage was given to the third frame and a significant weightage was given to first two frames too because both the frames depict a gun.

In Figure 5.2b the caption is ”a woman is boiling eggs in a pan”. We can observe that while predicting the word ”woman”, higher weightage was given to the first frame. While predicting the word ”boiling”, higher weightage was given to the third frame which depicts the eggs being boiled in water. While predicting the word ”eggs”, higher weightage was given to both second and third frames because both the frames depict eggs. While predicting the word ”pan”, more weightage was given to both first and second frames because both the frames depict a pan.

5.4.3 Adjusted Temporal Attention Analysis

Figures 5.3a and 5.3b illustrate the adjusted temporal attention weights, β_t given by Equation 4.16 for a particular word in the caption. The adjusted gate β_t is in the range $[0, 1]$. When $\beta_t = 1$ then complete visual information is considered using the context vector c_t, while when $\beta_t = 0$ then no visual information is considered and the next word is predicted using only the language context model.

In Figure 5.3a the caption is ”a man is shooting a gun”. The visual words in this caption are ”man”, ”shooting” and ”gun”. So it is expected that β_t values for these words must be higher than the rest and closer to 1. However we observe that the β_t values for ”man” (0.35), ”shooting” (0.45) and ”gun” (0.416) are relatively higher than the corresponding β_t values of the non-visual words but are all less

than 0.5.

In Figure 5.3b the caption is ”a woman is boiling eggs in a pan”. The visual words in this caption are ”woman”, ”boiling”, ”eggs” and ”pan”. We observe that the β_t values for ”boiling” (0.52) and ”eggs” (0.471) are relatively higher but for ”woman” (0.295) and ”pan” (0.263) are relatively lower than the corresponding β_t values of the non-visual words. Here also all the β_t values are closer to or lesser than 0.5.

We observe that the β_t values are not very contrasting to distinguish between visual and non-visual words. This suggests that the adjusted temporal attention weights are not being learnt properly.

5.4.4 Mean Aggregation Models Analysis

This section analyses the performance of the models discussed in Section 4.6. The hLSTMat-PCA 4.6.1 and hLSTMat-KMeans 4.6.2 variants are compared with the hierarchical LSTM with adjusted temporal attention (hLSTMat) model described in Section 4.5.3. The experiments are conducted on MSVD dataset and use Inception-v3 as the CNN encoder. The results are shown in Table 5.4.

Table 5.4: Comparison of mean aggregation models

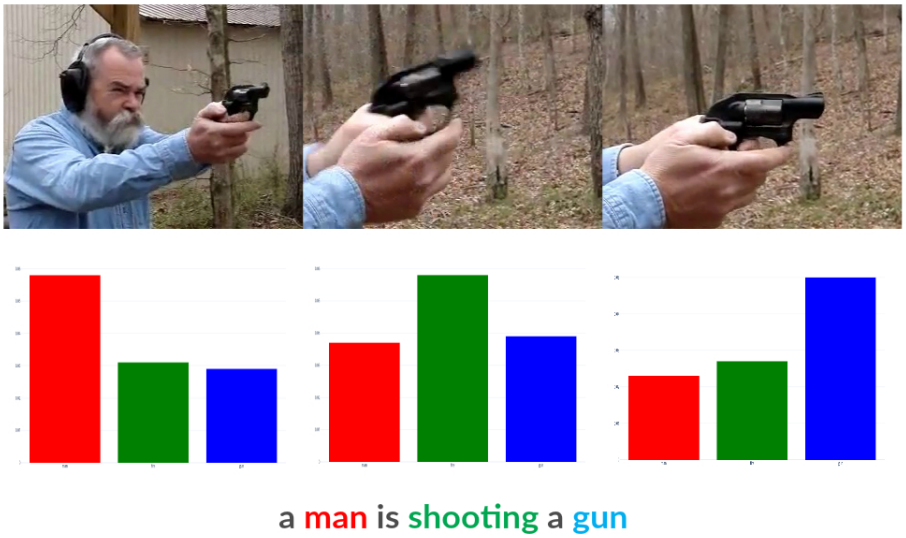
Decoder	BLEU@1	BLEU@2	BLEU@3	BLEU@4	METEOR
hLSTMat-PCA	81.5	71.0	61.9	51.3	33.7
hLSTMat-KMeans	82.9	72.4	63.4	53.3	34.1
hLSTMat	83.8	74.0	65.3	55.7	33.6

It is observed that hLSTMat-KMeans model performs better than hLSTMat-PCA model. There is a significant loss of information when PCA is performed to reduce the dimension of the video features from 2048 to 512, which in turn is

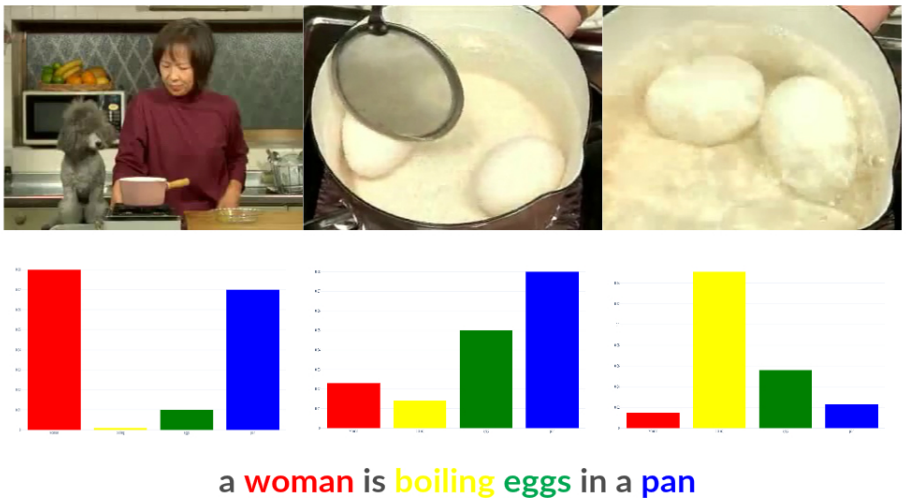
affecting the performance of the model.

Figure 5.4 shows temporal attention values for three frames while generating the word ”man” in the caption, for the video of a man shooting a gun in MSVD dataset, comparing hLSTMat, hLSTMat-PCA and hLSTMat-KMeans as decoder. It is observed that for hLSTMat-PCA, almost equal weightage was given for all three frames whereas for hLSTMat-KMeans and hLSTMat higher weightage was given to the first frame correctly.

Figure 5.5 shows the adjusted temporal attention values for a video of a man shooting a gun in MSVD dataset comparing hLSTMat, hLSTMat-PCA and hLSTMat-KMeans as decoder. The plot shows that neither of the models shows a clear contrast in values for visual and non-visual words.

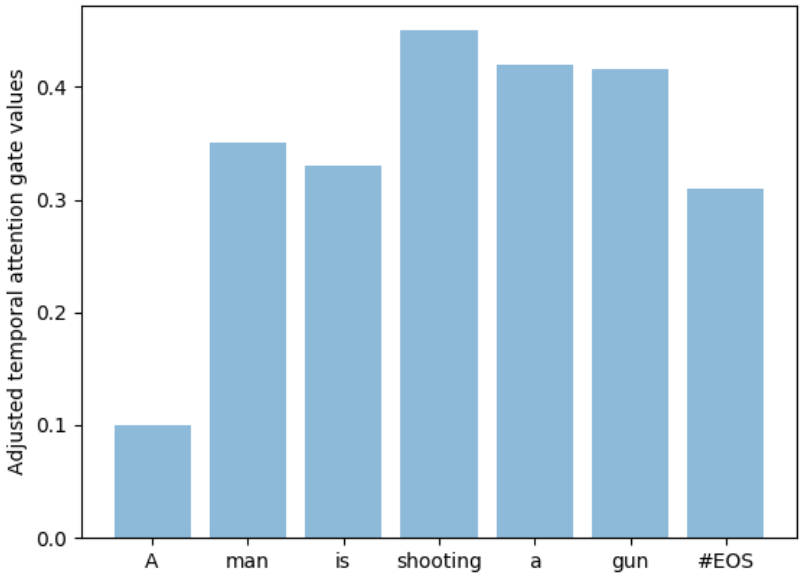


(a) Video of a man shooting a gun

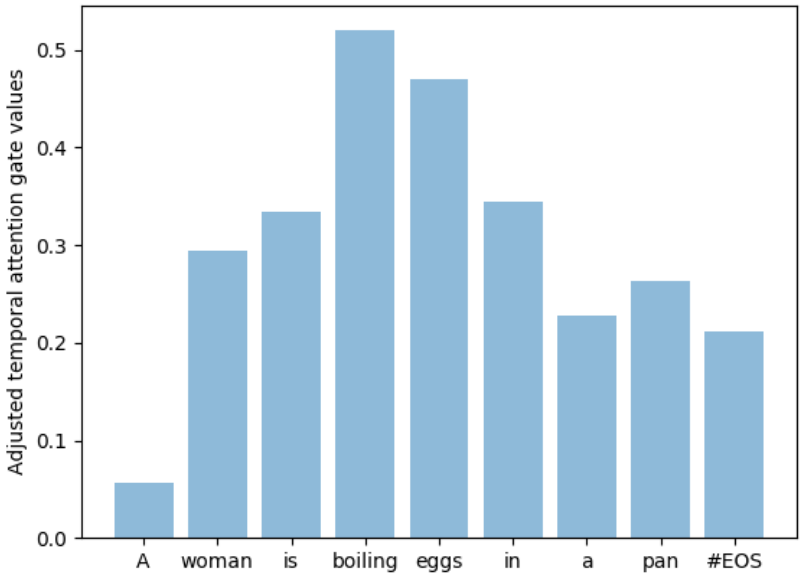


(b) Video of a woman boiling eggs in a pan

Figure 5.2: Temporal Attention plots for two videos in MSVD dataset using hLSTMat as decoder



(a) Video of a man shooting a gun



(b) Video of a woman boiling eggs in a pan

Figure 5.3: Adjusted temporal attention plots for two videos in MSVD dataset using hLSTMat as decoder

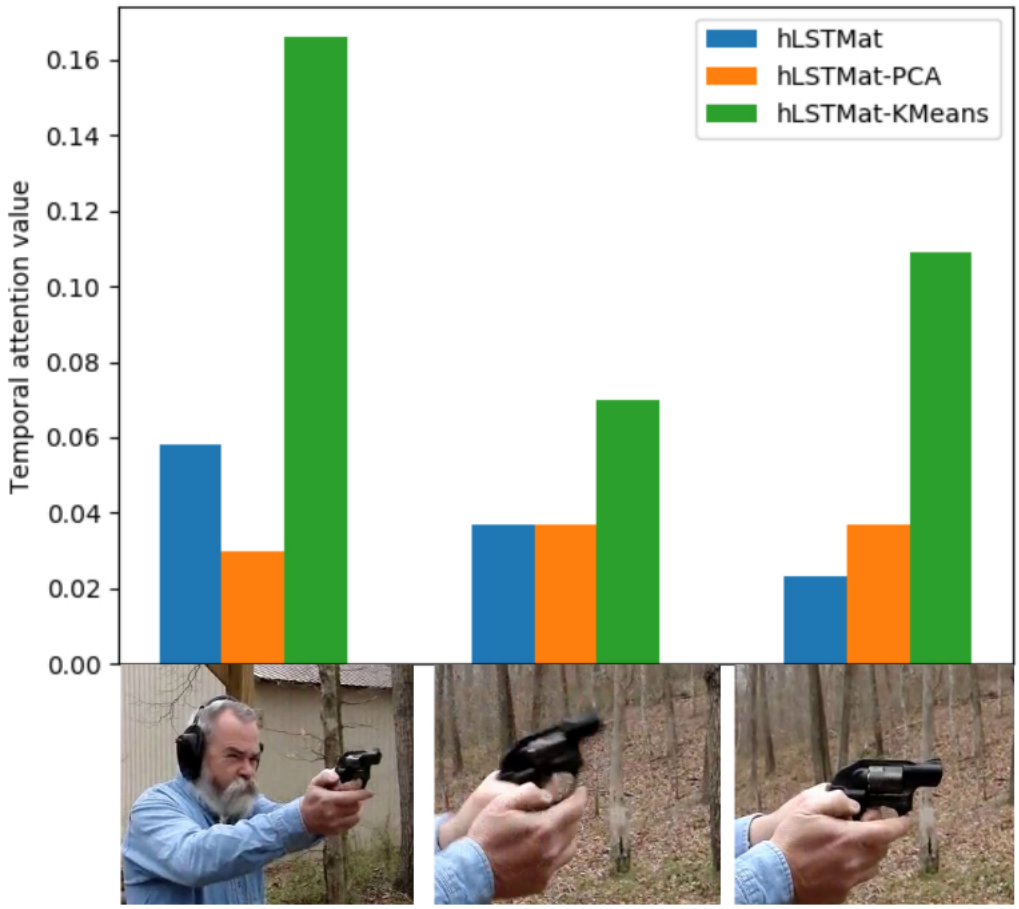


Figure 5.4: Temporal attention values for three frames while generating the word ”man” in the caption

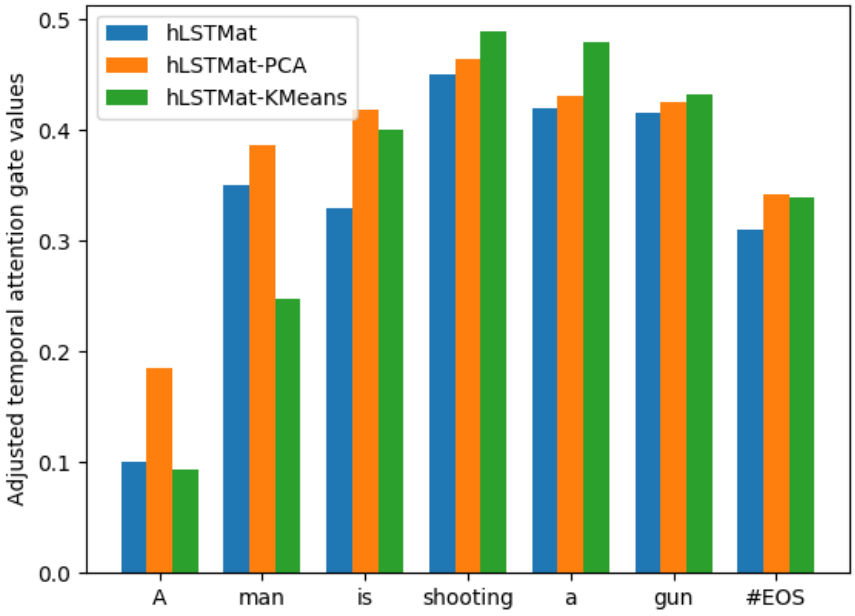


Figure 5.5: Adjusted temporal attention plots of a video in MSVD dataset comparing hLSTMat, hLSTMat-PCA and hLSTMat-KMeans as decoder

CHAPTER 6

Conclusion and Future Work

6.1 Conclusion

The Hierarchical LSTM with Adjusted Temporal Attention (hLSTMat) encoder-decoder framework for video captioning is explored. The model automatically decides whether to use language context information or visual information. When using visual information, the model enforces gradients from the visual features and decides where to attend. Experiments show that temporal attention and adjusted temporal attention improve the performance of the model and achieve state-of-the-art performance on MSVD dataset.

6.2 Future Work

The attention weights must further be fine tuned in the model. Currently, the model uses only spatial visual information. Incorporating the model with both temporal and spatial visual information could further improve the performance of the model.

REFERENCES

[1] Jingkuan Song, Lianli Gao, Zhao Guo, Wu Liu, Dongxiang Zhang, and Heng Tao Shen. Hierarchical LSTM with adjusted temporal attention for video captioning. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 2737–2743, 2017. doi: 10.24963/ijcai.2017/381. URL https://doi.org/10.24963/ijcai.2017/381.

[2] Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond J. Mooney, and Kate Saenko. Translating videos to natural language using deep recurrent neural networks. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 1494–1504, 2015. URL http://aclweb.org/anthology/N/N15/N15-1173.pdf.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012*, pages 1106–1114, 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014. URL http://arxiv.org/abs/1409.0473.

[5] Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher J. Pal, Hugo Larochelle, and Aaron C. Courville. Describing videos by exploiting temporal structure. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 4507–4515, 2015. doi: 10.1109/ICCV.2015.512. URL https://doi.org/10.1109/ICCV.2015.512.

[6] Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 3242–3250, 2017. doi: 10.1109/CVPR.2017.345. URL https://doi.org/10.1109/CVPR.2017.345.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90. URL https://doi.org/10.1109/CVPR.2016.90.

[8] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012. URL http://arxiv.org/abs/1212.5701.

[9] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *IEEE Conference on Computer Vision and Pattern*

Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015, pages 3156–3164, 2015. doi: 10.1109/CVPR.2015.7298935. URL https://doi.org/10.1109/CVPR.2015.7298935.

[10] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA.*, pages 311–318, 2002. URL http://www.aclweb.org/anthology/P02-1040.pdf.

[11] Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization@ACL 2005, Ann Arbor, Michigan, USA, June 29, 2005*, pages 65–72, 2005. URL https://aclanthology.info/papers/W05-0909/w05-0909.

[12] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL http://arxiv.org/abs/1409.1556.

[13] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2818–2826, 2016. doi: 10.1109/CVPR.2016.308. URL https://doi.org/10.1109/CVPR.2016.308.