Attention-Based LSTM Over Transformer Memory For Video Captioning

A Project Report

submitted by

AJITH KUMAR M

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

MASTER OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

APRIL 2019

THESIS CERTIFICATE

This is to certify that the thesis titled **Attention-Based LSTM Over Transformer Memory For Video Captioning**, submitted by **Ajith Kumar M**, to the Indian Institute of Technology, Madras, for the award of the degree of **Master of Technology**, is a bona fide record of the research work done by him under our 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.

Prof. C.Chandra Sekhar
Research Guide
Professor
Dept. of Computer Science and Engineering
IIT-Madras, 600 036

Place: Chennai

Date:

ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere gratitude to my guide Prof. C.Chandra Sekhar for the continuous support, patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better mentor for my M.Tech study.

I want to thank Prof. Hema Murthy and Dr. Mitesh Khapra for their excellent course in Pattern Recognition and Deep Learning. I would also thank my fellow labmates Rupam Ojha and Ujjal Kumar Dutta for welcoming me into the Speech and Vision Lab. I thank Ms. Jyostna Devi Bodapati for her timely insights. I appreciate the DON lab folks for sharing the GPU servers. I also thank my friends Jeshuren, Mahesh, Ganesh, Murali, and every one of my classmates who have directly on indirectly helped me in multiple ways. Finally, yet importantly I thank my parents for everything.

ABSTRACT

KEYWORDS: Video Captioning; Seq2Seq; LSTM; Language Model; Transformer; Attention.

The task of visual captioning is one of the applications of deep learning. The report discusses a few approaches to visual captioning and starts with implementation of one such approach to video captioning called aLSTMs. Later it discusses about the effect of coupling a language-model based loss to improve the performance of the system. It also contains experiments with video captioning on the transformer framework which was primarily developed for translation tasks. Finally, it show how a hybrid model with LSTM and Transformer outperforms most of the earlier approaches and achieves a near State-Of-The-Art performance with addition of the language-model based loss.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
GRU	Gated Reccurent Unit
CNN	Convolution Neural Network
SIFT	Scale Invariant Feature Transform
HOG	Histogram of Oriented Gradients
HOF	Histograms of Optical Flow
MBH	Motion Boundary Histogram
BPTT	BackPropagation through time
C3D	Convolutional 3D
aLSTMs	Attention-based LSTM model with semantic consistency
LM	Language Model
NLTK	Natural Language Tool Kit
t-SNE	t-Distributed Stochastic Neighbor
GloVe	Global Vectors for Word Representation
MSVD	Microsoft Video Description Corpus

CHAPTER 1

INTRODUCTION

1.1 Introduction

The task of captioning images and videos is important for a vast range of applications. An efficient method to generate captions automatically will assist people who are not able to watch the videos, and also help them discern their surrounding. The tremendous progress in deep learning research has made it possible for machines to solve a variety of challenging tasks such as visual classification and object detection, with near-human level performance. Visual captioning is one such challenging task where to generate a proper caption; the system has to detect the visual concepts in the visual space, understand the interactions and then generate captions that are syntactically and semantically correct. Further, it is also hard to estimate the degree of correctness for a generated caption since multiple captions might happen to reasonably agree for a single image or a video for that matter. An example video and it's corresponding caption is shown in fig. 1.1.

A man is cooking

Figure 1.1: Video captioning example.

The task of video captioning is analogous to machine translation where instead of mapping a sequence of words from one language to another, we map a sequence of frames that make up a video to a sequence of words that describes the video. Therefore, in theory, methods that work for machine translation should work for video captioning as well. Conventionally Recurrent Neural Networks(RNNs) are a good choice for

these sequence to sequence learning tasks, and such models are called Seq2Seq models. More advancements in sequence modeling have given us better RNN architectures such as Long Short-Term Memory(LSTM)[11] and Gated Recurrent Unit(GRU)[12], which showed exceptional performance with any task involving temporal data. This report discusses different such approaches for video captioning and reports the effect of using an external language model to help improve the performance of the video captioning system on the Microsoft Research Video Description(MSVD) and the Microsoft Research Video-To-Text (MSR-VTT) datasets. It also describes the transformers architecture proposed in [13], that was primarily developed for machine translation, and reports how hybrid version of the same has improved the performance of video captioning systems.

1.2 Organization of Report

The rest of the report is organized as follows. Chapter 2 presents the overview of required basic blocks such as CNN, RNN, and LSTM for feature extraction and sequential modeling. It also discusses about sentence vectors and Language-Models. Chapter 3 presents various existing approaches to video captioning. The aLSTMs approach is discussed in detail, and other approaches that were used for comparison are also discussed briefly. Experimental studies on the aLSTMs approach are presented in Chapter 4, and a new approach for augmenting the objective function for the video captioning system is presented in Chapter 5. Chapter 6 explains how the Transformer framework could be used for video captioning and provides experimental studies on the same. In Chapter 7, a novel hybrid that integrates Transformer and LSTM is introduced. The penultimate chapter is about the analysis of the captions generated by different methods with their ground truths. The final chapter summarises the report and presents the conclusion.

CHAPTER 2

Deep Learning Models for Feature Extraction and Sequential Patterns

2.1 Convolutional Neural Network

The convolutional neural network(CNN) is a feedforward neural network that has proven to be effective in the area of image and object recognition. It consists of the repeated arrangement of convolution, pooling and dense layers with nonlinearities to capture a deeper representation of the image, which can then be used for the classification of the image. The convolution layer contains a set of learnable convolution filters which generate feature maps when convolved around an image. When multiple such layers are stacked on top of one another, it generates a deeper feature representation of the image. The final layers are fully connected dense layers that are responsible for classification. Since the whole network is connected end-to-end, the feature extraction and classification are learned together from the final classification loss alone. Fig. 2.1 depicts a typical CNN classifier. A pre-trained CNN can then be used for feature extraction by

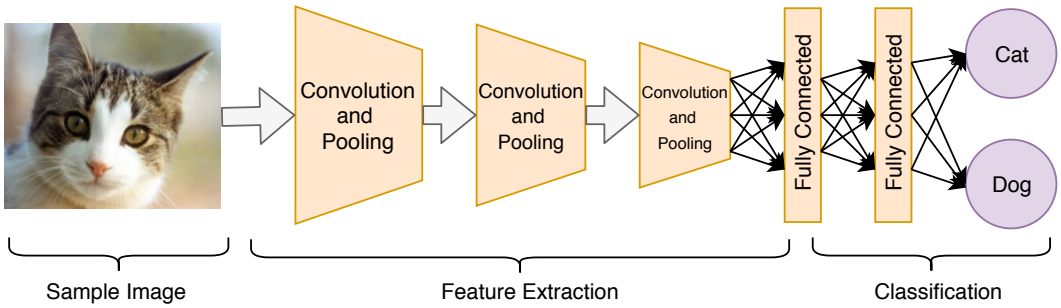

Figure 2.1: CNN classifier.

using the activations of an appropriate fully connected layer. The deeper representations learnt through CNNs are believed to be superior to the classical SIFT(Scale Invariant Feature Transform) and HOG(Histogram of Oriented Gradients) features. Today, with the help of dropout regularization and residual connections, deeper CNNs can be built

without the problem of vanishing gradients and overfitting. Some of the popular CNN architectures are VGG, Inception, and ResNet.

2.2 Recurrent Neural Network

Recurrent neural networks(RNN) address the problem of modelling temporal dependencies in data. They have an internal state or memory and recurrent connections to learn and model sequential data. Fig 2.2 shows an RNN unfolding for input sequence X and output sequence O. Despite the fact that RNNs were good at modelling sequential data, such as handwriting and speech, they couldn't handle long temporal dependencies in the data. This was mainly due to the problem of vanishing gradients in RNNs [14]. Since RNNs were usually trained by back-propagation through time(BPTT), they were unfolded into a feedforward neural network with many layers. When the gradient flows back through time steps, it tends to explode or vanish, just as in a deep feedforward network. Therefore modelling with long sequential data was difficult with RNNs.

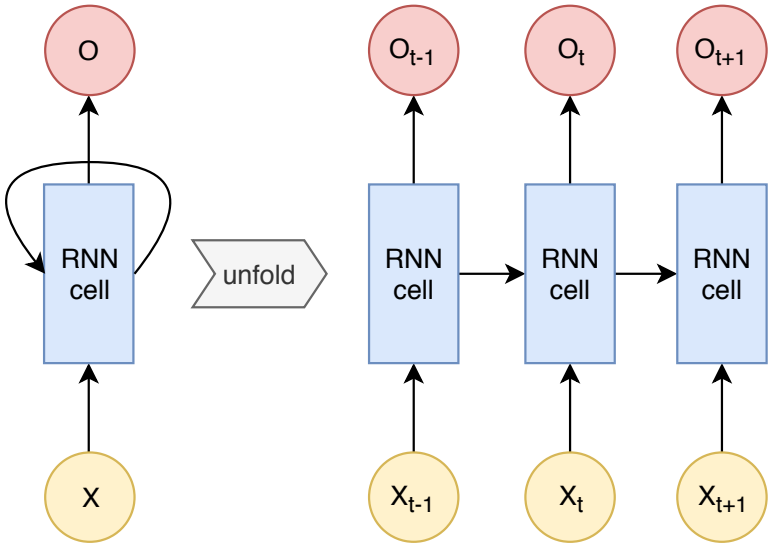

Figure 2.2: RNN unfolding.

2.3 Long Short-Term Memory

To address the problem of vanishing gradients with RNNs, the Long Short-Term memory networks(LSTMs) were proposed [11]. They were a special kind of RNNs, capable

of learning long-term dependencies in temporal data without any vanishing gradient issues. It comes with separate cell state and hidden state which acts as memory, and dedicated gates to regulate the flow of data in these states. Fig. 2.3 shows a typical LSTM cell. Here c_t and h_t are the cell state and hidden state of the LSTM and f_t , i_t and o_t are the forget gate, input gate and output gate respectively. σ and ϕ represent logistic sigmoid and tanh nonlinearities. The equations that represent Fig. 2.3 are given below where W_x and b_x are trainable parameters. Note that here * represents element-wise multiplication.

$$f_t = \sigma(W_f.[h_{t-1}, d_t] + b_f) \tag{2.1}$$

$$i_t = \sigma(W_i.[h_{t-1}, d_t] + b_i) \tag{2.2}$$

$$g_t = tanh(W_g.[h_{t-1}, d_t] + b_g) \tag{2.3}$$

$$c_t = f_t * c_{t-1} + g_t * i_t \tag{2.4}$$

$$o_t = \sigma(W_o.[h_{t-1}, d_t] + b_o) \tag{2.5}$$

$$h_t = o_t * tanh(c_t) \tag{2.6}$$

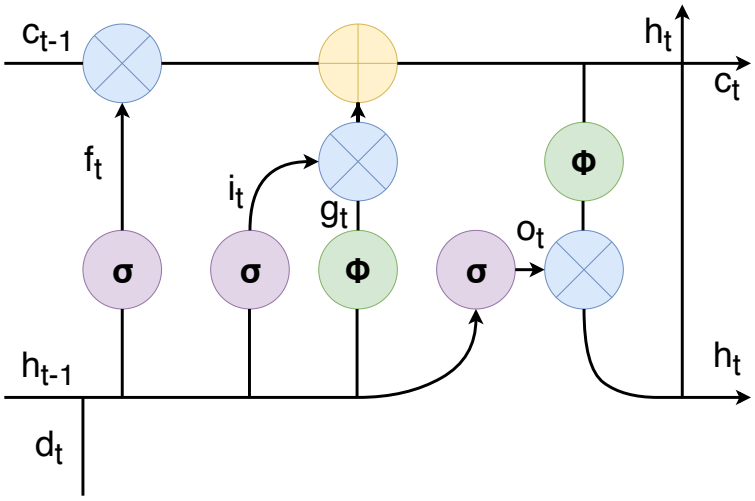

Figure 2.3: The LSTM cell.

A slight variation of the LSTM called the Gated Recurrent Unit(GRU) was introduced in [12]. It uses a single update gate in place of the input and forget gates of the LSTM. It is simpler than the standard LSTM and has fewer number of parameters, which resulted in faster training time than LSTMs.

2.4 Sentence Vectors

Converting a variable length natural sentence into a fixed size feature vector is known as sentence vectorization. Such sentence vectors should arguably capture the essence of the sentence in the feature space. Sentences that are more relatable and synonymous should also be closer in the feature space. A naive method for sentence vectorization is the Bag-of-Words(BoW) method. Here each component of the feature vector is just the occurrence of a specific word in the sentence. Since it's vocabulary is limited to the words in the training data, a novel word in test data might be missed out in the vectorization process. Another approach to sentence vectorization is by aggregating the word2vec embeddings (learnt from a large corpus) of all the words in the sentence. The drawback of this technique is that it simply ignores the word order in the sentences. As far as BoW and word2vec are considered "Ram killed Ravan" and "Ravan killed Ram" are just the same.

This issue can be overcome by using an LSTM or GRU to model the sentence vectors since they are capable of learning word dependencies in natural sentences. A pre-trained LSTM can come up with an effective representation of the sentences [15].

2.5 Language Model on One Billion Word Benchmark [1]

A large language corpus in English was released in 2013 called the One Billion Word Benchmark [16]. It contains 1B words and has a vocabulary of 800K words. Several approaches have been tried to develop a language model using this dataset. One such approach is a model hybrid between character CNN[17] and a deep LSTM. This is the best language model on this dataset so far. The sentences in this dataset are shuffled before training so that models can ignore the context and focus on sentence-level language modelling. Since this is an LSTM based language model, for a sentence fed into this model, the final state of the LSTM captures the latent meaning of the sentence and it can also be treated as the sentence vector for the sentence.

2.6 Summary

In this chapter, we discussed an approach for feature extraction from images using CNNs. We also discussed approaches to model sequential data using RNNs and one of its popular variant called the LSTM. Later we discussed about sentence vectors and a popular language model. Next chapter discusses approaches to Video Captioning using the methods mentioned here.

CHAPTER 3

Approaches to Video Captioning

3.1 The End-to-End Framework

In order to generate captions from visual data, the system has to understand the components in the visual data and then generate captions based on its understanding. Therefore the whole video is supposed to be processed before the system starts to generate any caption. This is how any human will go about describing an image or video. These types of models are called encoder-decoder models in deep learning. In such models, the data is first encoded into a fixed dimension feature vector, and then this feature vector is used for decoding the desired output from the data. In our case, the video should be encoded into a feature vector, and the captions shall be decoded from that vector. One such popular encoder-decoder based framework is the Sequence-to-Sequence(Seq2Seq) framework [18]. Fig.3.1 shows a typical Seq2Seq architecture using LSTM.

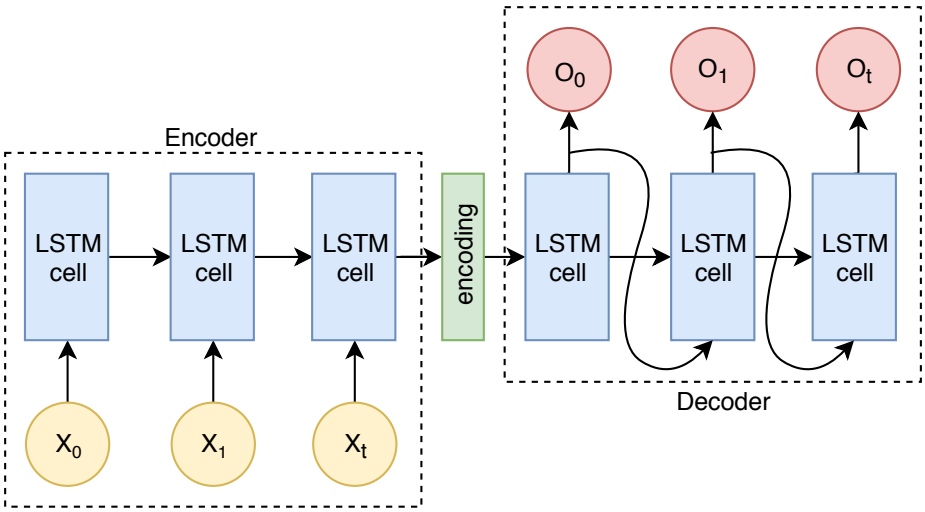

Figure 3.1: Seq2Seq.

Without loss of generality, any sequential data conversion can be modeled efficiently in the Seq2Seq framework. Usually, for a Seq2Seq model, an LSTM is used to encode the input sequence into a fixed size vector, which is usually the cell state(or hidden state) of the final time step of the encoder. This encoded vector will then be used by the

decoder(usually the decoder state is initialized with the encoded vector) to generate the desired output sequence. Such a model is connected end-to-end and thus can be trained using the gradient descent with BPTT.

3.2 Video Captioning with Attention

Video captioning is more related to the Seq2Seq framework since the input is also a sequence(of frames). A video can be considered as a 3D image(the third dimension being temporal) and be encoded using C3D [19]. This is not efficient enough in learning temporal dependencies. Another method is to encode the video using an LSTM. The CNN representation vectors(global visual vector) for the frames(images) of a video are used as sequential data for efficient encoding by an LSTM. The encoded vector, which is usually the final hidden state of the LSTM is then used by the decoder to generate captions. Since it is compute expensive to find the visual vector for each individual frames, a frame is picked at regular intervals or a uniform sampling is done to select a subsequence of frames, and then compute the visual vectors for those frames. To improve the performance of the system, we could incorporate the **attention** mechanism in the model [20]. Attention enables the decoder to focus on the relevant parts of the input sequence as needed. It is achieved by calculating the alignment weights for each time step of the encoder and then feeding the weighted sum of the encoder outputs to the decoder. Usually, the attention weights(or alignment vector α) are calculated as a function of the decoder state and the encoder hidden states and normalized with a softmax.

$$e_{jt} = f_{ATT}(s_{t-1}, h_j) \tag{3.1}$$

$$\alpha_{jt} = \frac{exp(e_{jt})}{\sum_{j=1}^{M} exp(e_{jt})} \tag{3.2}$$

$$c_t = \Sigma_{i=0}^{M} \alpha_i h_i \tag{3.3}$$

Here e_{jt} is the unnormalized alignment weight, h_j is the encoder output at frame j, s_t is the decoder state at time step t, f_{ATT} is a feedforward neural network transformation, M is the number of frames and α_{jt} is the normalized alignment weight. c_t is

the context vector fed to the decoder at time step t. In this way, the decoder can focus on relevant frames while decoding the caption. Many approaches were proposed to improve the attention mechanism, such as review module [21] and bottom-up attention model [22]. In the next section, we will look into training the video captioning models with semantic consistency as proposed in [2].

3.3 Video Captioning with Semantic Consistency [2]

In the previous approach, we discussed training encoder-decoder models with attention for video captioning. In these models, the objective is to minimize the sequential cross-entropy loss in the decoder. To further enhance the error signals, we could design an objective function such that it also considers the correlation between the sentence semantics and the visual content. To achieve this, we need a vector representation of the corresponding caption and project it into a common feature space with the vector representation of the video. To get a vector representation of the caption, we could average the individual word embeddings of the caption from the embeddings learned by training the model. This sentence vector can then be projected to the same dimension as that of the visual vector by using a feed-forward network. The new loss function is the mean-squared-error loss between the two vectors. The objective function is as follows.

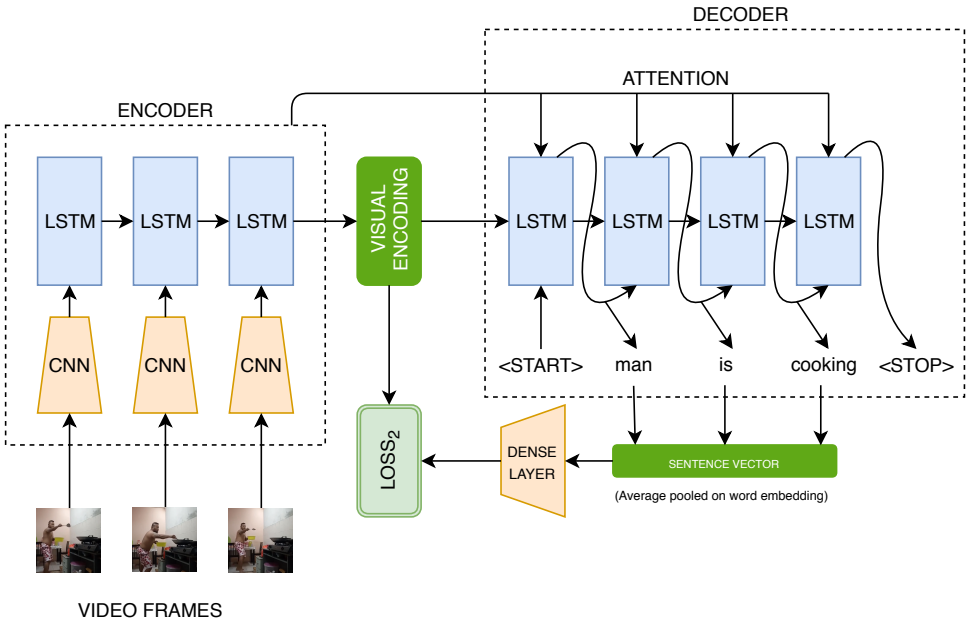

Figure 3.2: aLSTMs architecture.

$$Loss_1 = -\Sigma_{t=0}^{N_d}(log(P(d_t|d_1, d_2 \ldots, d_{t-1}))) \tag{3.4}$$

$$Loss_2 = \frac{\Sigma_{i=0}^{n-1}(x_i - d_i)^2}{n} \tag{3.5}$$

$$Loss_{total} = \lambda Loss_1 + (1 - \lambda)Loss_2 \tag{3.6}$$

Here d_i denotes the i^{th} word of the sentence and N_d is the number of words in the caption. x is the visual encoding vector from the encoder, d is the sentence vector projected on the same dimentional space as that of the visual encoding vector, n is the dimention of the visual encoding vector, $Loss_1$ in the sequential cross-entropy loss from the decoder, $Loss_2$ is the semantic consistency loss and $Loss_{total}$ is final loss function that we need to minimize. λ is a hyper-parameter which acts as a trade-off between the two losses. This novel framework has been named as attention-based LSTM model with semantic consistency(aLSTMs) shown in Fig. 3.2.

3.4 Other Popular Approaches

3.4.1 Soft-Attention (SA)[3]

SA uses two types of features: frame-level features extracted from pre-trained CNN; and video level features which are extracted by a 3DC. This 3DC vector is concatenated with a set of descriptors including HOG, Histograms of Optical Flow (HOF), and Motion Boundary Histogram (MBH). Furthermore, a weighted attention mechanism over the frame-level features is employed to enhance sentence generation.

3.4.2 Video Paragraph Captioning Using Hierarchical Recurrent Neural Networks (p-RNN)[4]

This approach uses an hierarchical-RNN framework for describing a long video with a paragraph consisting of multiple sentences. This framework consists of two parts. A sentence generator which produces single short captions corresponding to specific time intervals and a paragraph generator which takes the sentential embedding of the first

part as input and uses another RNN to output the paragraph.

3.4.3 Hierarchical Recurrent Neural Encoder for Video Representation with Application to Captioning [5]

In order to make use of temporal information of videos, this approach introduces a hierarchical recurrent neural encoder (HRNE) by utilizing a soft attention mechanism to generate captions for videos

3.4.4 Hierarchical LSTM with Adjusted Temporal Attention for Video Captioning (hLSTMat) [6]

hLSTMat is an encoder-decoder approach that integrates a hierarchical LSTMs, temporal attention and adjusted temporal attention to automatically decide when to make good use of visual information or when to utilize sentence context information, as well as to simultaneously considering both low-level video visual features and language context information.

3.4.5 Video Captioning with Transferred Semantic Attributes (LSTM-TSA) [7]

LSTM-TSA architecture explores both video representations and semantic attributes for video captioning. It also discusses on how to mine attributes from images and videos and how to fuse them in an elegant manner for enhancing sentence generation.

3.5 Summary

In this chapter we had discussed multiple approaches to video captioning. Most of these approaches have achieved near state-of-the-art performance in the video captioning domain. In the next chapter, we shall discuss about the implementation and experiments of the aLSTMs approach.

CHAPTER 4

Experiments on aLSTMs [2]

4.1 Dataset

4.1.1 MSVD [8]

The Microsoft video description corpus(MSVD) consists of around 122K descriptions, for 2089 video clips but only 1970 of those video clips are available in public domain. Also out of 122K descriptions, only 85K are English descriptions. This dataset is split into 1,200, 100 and 670 clips for training, validation and testing respectively.

4.1.2 MSR-VTT [9] [10]

The Microsoft Research - Video to Text (MSR-VTT) is a large-scale video description dataset with 10K video clips and around 200K English clip-sentence pairs. The videos are further split into 20 different catagories. This dataset is split into 6,513, 2,990 and 497 clips for training, validation and testing respectively.

4.2 Evaluation Metrics

Various methods have been proposed to evaluate the degree of correctness and quality of the generated captions. While human evaluation tends to be the best and most accurate, it also happens to be tedious and requires a tremendous amount of labor. BLEU, METEOR, CIDEr, and SPICE are some of the commonly used automatic metrics. BLEU compares the number of common n-grams between the generated text and ground truth. METEOR compares unigram precision and recall by also allowing stemmed words and similar words based on WordNet. CIDEr is similar to BLEU but weighs the words with their TF-IDF scores. BLEU-4 and METEOR are the ones that are widely used for evaluation.

4.3 Implementation details

4.3.1 Dataset preparation

The MSVD dataset was split into 1200, 100 and 670 clips as train, validation and test set respectively. The descriptions were converted into lowercase and were tokenized using wordpunct_tokenizer from the NLTK[23] toolkit. Thus the punctuations were removed from the descriptions. Further, descriptions of more than 32 words were removed from the dataset, with the assumption that they are noisy. All the descriptions were appended with <START> and <END> as start and end symbol respectively. For the video clips, equally-spaced 28 frames out of the first 360 frames were extracted for feature extraction.

4.3.2 Feature extraction

The pre-trained Inception-v3 model[24] was chosen for feature extraction from the frames. This model was trained on ImageNet Large Visual Recognition Challenge dataset[25]. For each frame, we obtain a 2048-dimensional feature vector from the last avg-pooling layer of the Inception-v3 model.

4.3.3 Training

The model architecture mentioned in [2] was implemented with Tensorflow [26]. The LSTM unit size was set to 1024 and the word embedding size was set to 512. The minibatch size was set to 1024 and dropout probability was set to 0.5. The Adam[27] approach was adopted to optimize the objective function with a learning rate of 0.001, β_1 and β_2 was set to 0.9 and 0.99 respectivly. Gradient clipping(norm-based) was also adopted at 10. The trade-off parameter λ was set at 0.9 and this model was trained for 200 epochs with a patience of 50 epochs. The results of the thus trained model are compared with the other techniques on MSVD dataset in Table 4.1

From table 4.1, it is evident that aLSTMs has performed better than some of the mentioned approaches. It is to be noted that the other approaches mentioned in 4.1 were proposed before the aLSTMs approach.

Table 4.1: Performance of aLSTMs compared with other techniques on the MSVD dataset.

Model	**B@1**	**B@2**	**B@3**	**B@4**	**METEOR**	**CIDEr**
HRNE[5]	78.4	66.1	55.1	43.6	32.1	-
SA[3]	-	-	-	40.3	29.0	51.7
HRNE-SA	79.2	66.3	55.1	43.8	33.1	-
p-RNN(C3D+VGGNet) [4]	**81.5**	**70.4**	**60.4**	49.9	32.6	-
aLSTMs	79.7	68.3	59.7	**50.9**	**34.5**	**72.9**

4.4 Summary

In this chapter, we have discussed about the experimental studies of the aLSTMs approach to video captioning. This approach showed how a simple addition of semantic consistency loss to the objective function of the model could boost its performance. In the next chapter we shall discuss about a novel method that uses an Language-Model based loss for training video captioning models.

CHAPTER 5

Video Captioning with External Language Model for Sentence Vectorization (LSTM+LM)

5.1 Motivation

In the previously discussed model, the sentences were vectorized using the word embeddings learnt while training the model. This technique will induce instability while training since the targets for the encoder output are dependent on the embeddings which change for every weight update during training. Moreover, this sentence vector is not significantly powerful as it is learnt from the descriptions corpus alone. If we could have an effective static sentence vector representation for a caption, then this vector could be the partial target for the encoder. In other words, we expect the encoder to encode the video into a vector such that the video encoding resembles the sentence vector of the sentence that describes the video.

5.2 Proposed Approach

In section 2.5, an LSTM based language model that could generate high-quality sentence embeddings has been discussed. These embeddings are extracted from a model trained on a large external corpus. The t-SNE plot of the sentence vectors obtained from the above-mentioned language model for 10 sample videos are shown in Fig. 5.1. Each of the videos has multiple captions and it can be seen that captions of a video form a cluster in the t-SNE plot. In other words, different captions describing the same video are closer in the sentence vector space. Some sample captions for the videos corresponding to the sentence vectors plotted in Fig 5.1 are shown in Table 5.1. It can be observed that the captions of Video 9 and Video 10 are similar in meaning and therefore similar in vector space as well.

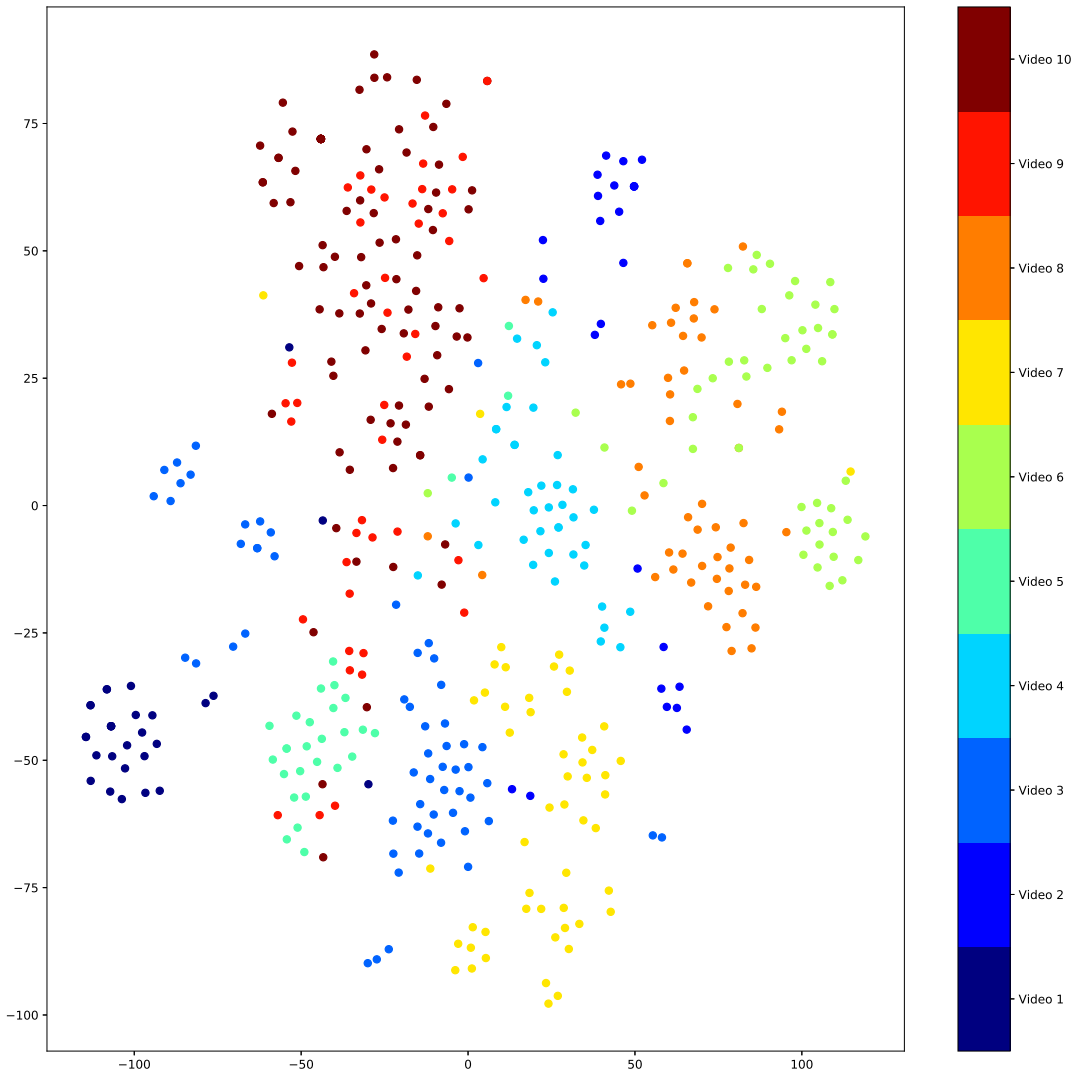

Figure 5.1: t-SNE plot of sentence vectors.

Table 5.1: Captions for sentence vectors in Fig. 5.1

Video ID	Captions
1	the lions are watching television a male and a female lion are watching tv a couple of lions are sitting on a couch watching tv
2	a man is shopping drink bottles a man is picking a bottle from a super market the man paid for the beer at the counter
3	a jet is flying over a crowd an airplane is flying in the sky with a crowd of people watching a jet plane is speeding through a cloudy sky
4	the man is lifting weights the man lifted a lot of weights a body builder is doing exercises
5	two persons are dancing in the stage the men are performing on stage two men are robot dancing on stage
6	a guy is chopping garlic the man is smashing garlic a person is cutting a slice of onion
7	a man is showing stunt with his bike a man riding a dirt bike crashes a motor cycle crashes in the sand
8	a person is frying something in a frying pan a man is preparing food meat is being cooked on a grill
9	a man is playing a guitar a band is performing on a stage the man sang with the band on stage
10	two men are playing electric guitars two men are playing guitar on a television show two men are playing electric guitar and bass

The LSTM unit size is adjusted so that the encoder output dimension matches the dimension of the sentence embedding, thus eliminating the need for a feedforward network unlike aLSTMs. A loss function, $loss_2$ is derived from the L2 norm of the difference between the encoder output and sentence vector.

$$Loss_2 = \|x - p\|_{L2} \tag{5.1}$$

Here x is the visual encoding vector, and p is the sentence vector of the ground truth caption obtained from a pre-trained language model. $loss_1$ is the sequential cross entropy loss from the decoder. The objective function is now similar to that of the aLSTMs model. Fig 5.2 shows the architecture of the proposed approach.

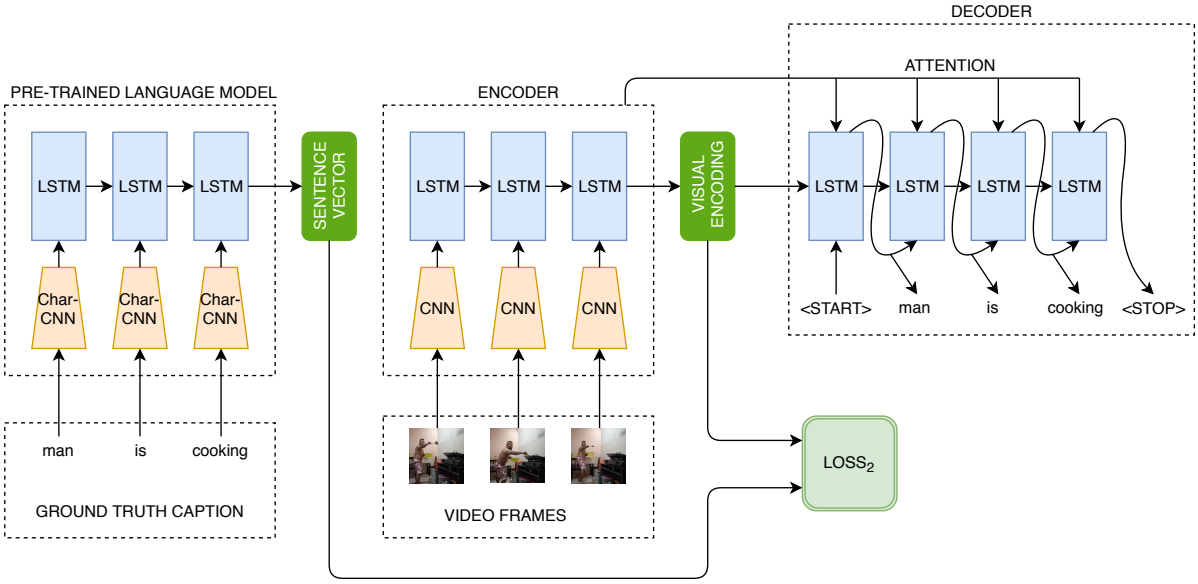

Figure 5.2: Proposed framework using a pre-trained language model

5.3 Implementation

The LSTM unit size was set to 512, and the word embedding size was set to 256. The word embeddings were randomly initialized. GloVe initialization was also tried, but no significant improvement was seen. The minibatch size was set to 1024 and dropout probability was set to 0.5. The Adam approach was adopted to optimize the objective function with a learning rate of 0.001, β_1 and β_2 were set to 0.9 and 0.99 respectively. λ was set at 0.9 and this model was trained for 200 epochs with an early stopping patience

of 50 epochs. For inference, the beam search decoder of beam size of 5 was used. The results of the thus trained model are compared with aLSTMs implementation and the current state-of-the-art models for the MSVD dataset in Table 5.2

Table 5.2: Results of LSTM+LM compared with aLSTMs on MSVD dataset

Model	**B@1**	**B@2**	**B@3**	**B@4**	**METEOR**	**CIDEr**
aLSTMs	79.7	68.3	59.7	50.9	**34.5**	72.9
Proposed Approach(LSTM+LM)	**81.8**	**70.7**	**61.3**	**51.7**	33.5	**77.0**

It is evident from Table 5.2 that the approach proposed in this chapter has performed better in most of the metrics than the aLSTMs framework discussed in the previous chapter. This is owing to the fact that external sentence vector embeddings in LSTM+LM are better representations than simple bag-of-words embeddings of aLSTMs.

5.4 Summary

A novel method to video captioning was introduced in the chapter and experimental studies were performed on the same. This approach utilizes an external language-model based loss for training and due to which, the model perform better than the aLSTMs approach. In the next chapter, we shall discuss about a new framework that was developed for machine translation tasks and experiment on this framework for the video captioning domain.

CHAPTER 6

Video Captioning with Transformers

6.1 Introduction

The LSTM based architecture is a popular choice for modeling the machine translation task. In recent trends, most of the proposed Seq2Seq architectures use an attention mechanism to improve their performance. The Seq2Seq architecture and attention mechanism have been discussed in section 3.1 and 3.2 respectively. The attention mechanism is thus believed to provide a significant boost to LSTM based architectures. In any RNN/LSTM based Seq2Seq framework, the flow of information is usually long and might lead to loss of information. For example, Fig. 6.1 shows how the word "dog" is observed at timestep t_0 and emitted at timestep t_4 . So naturally, the information of "dog" needs to be propagated for at least four timesteps before it could be forgotten. Attention fixes this issue by providing a direct shortcut to the encoder state for the decoder. A novel architecture called "Transformer" was introduced in the paper 'Attention Is All You Need'[13]. As the title indicates, it uses the attention mechanism alone to model sequence to sequence data without any RNN based structure in the model. It has also shown better performance than the RNN based counterparts in specific language translation tasks.

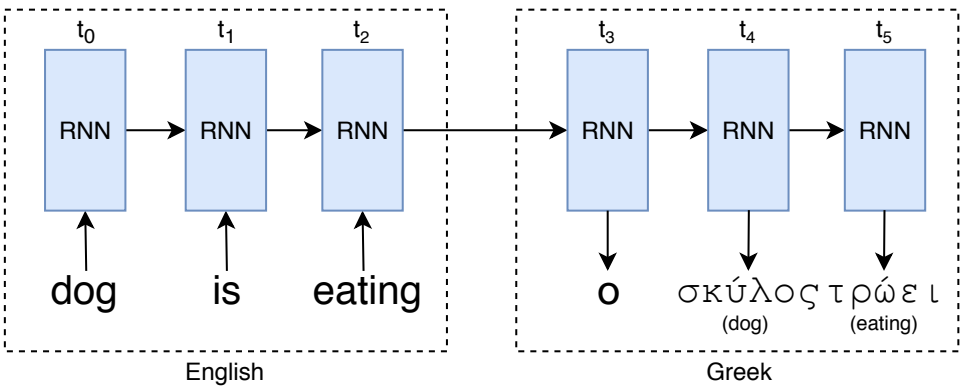

Figure 6.1: English to Greek translation using RNN.

6.2 Transformer Architecture

6.2.1 Scaled Dot-Product Attention

The Transformer uses a particular type of attention technique known as scaled dot-product attention. The equation for scaled dot-product attention is given in Eq. 6.1.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{6.1}$$

Here Q is a matrix of attention **queries** with each row i in the matrix representing a query q_i. K is a matrix of **keys** and V is a matrix of **values** with each row j in the matrix representing a key k_j and value v_j respectively. The number of keys and values are always equal and has a one-to-one correspondence, and let that be n_k. d_k is the dimension of a key vector. Let d_q and d_v be the dimensions of the query vector and value vector respectively. For simplicity, let us assume that the dimension of query, key and the value vector are same, i.e. $d_q = d_k = d_v$, and consider a single query vector q_0 with dimension $1\text{x}d_q$. The product of q_0 and K^T results in a vector of dimension $1\text{x}n_k$. The j^{th} element of this vector is the dot product of vector q_0 and k_j.

$$a = softmax(\frac{QK^T}{\sqrt{d_k}}) \tag{6.2}$$

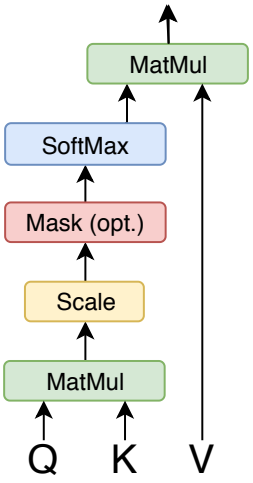

Figure 6.2: Scaled dot-product Attention.

Since dot product is an unnormalized measure of similarity between two vectors,

each element j of this vector represents the unnormalized similarity weights of q_0 with k_j. The weights are then normalized with a softmax function to get normalized attention weights a of the query vector q_0 over keys K. Note that the unnormalized weights are scaled by a factor of $\frac{1}{\sqrt{d_k}}$ before normalizing. Finally, the attention weights are multiplied with V to obtain a weighted sum of the values. This can be generalized to multiple queries in which case, the output of scaled dot-product attention will be a matrix of dimension n_qxd_v, with each row i representing the attention outputs of query q_i over values V. Fig. 6.2 illustrates scaled dot-product attention.

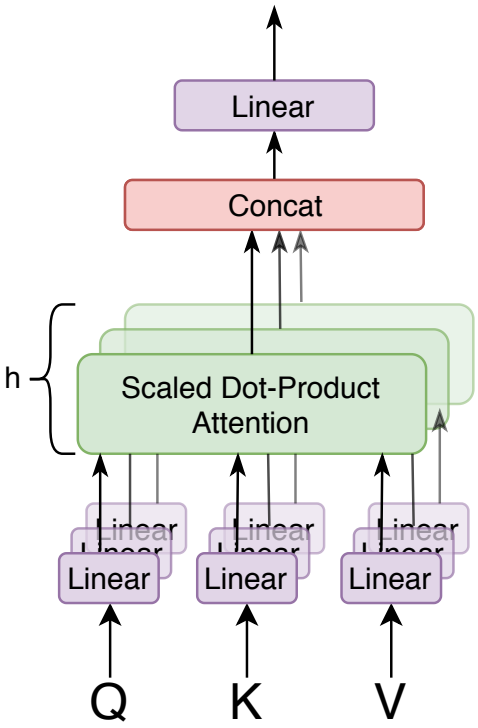

Figure 6.3: Multi-Head Attention.

6.2.2 Multi-Head Attention

Instead of using a single scaled dot-product attention mechanism, it would be beneficial to go with multiple such attention mechanisms. This could be achieved by projecting Q,K, and V by using linear transformations with learnable parameters, each to h different subspaces of dimension d_k,d_k and d_v respectively. For each of the h subspaces, we would apply scaled-dot product attention and collectively get h attention output matrices. These attention outputs are concatenated and passes through a final linear transformation, such that the dimension of the final output is same as that of Q. The

Equations for multi-head attention are as follows.

$$MultiHead(Q, K, V) = Concat(head_1, head_2, \ldots, head_h)W^O \tag{6.3}$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{6.4}$$

Here W^O,W_i^Q,W_i^K,W_i^V are matrices with trainable parameters. Fig 6.3 shows the multi-head attention mechanism.

6.2.3 Transformer Block

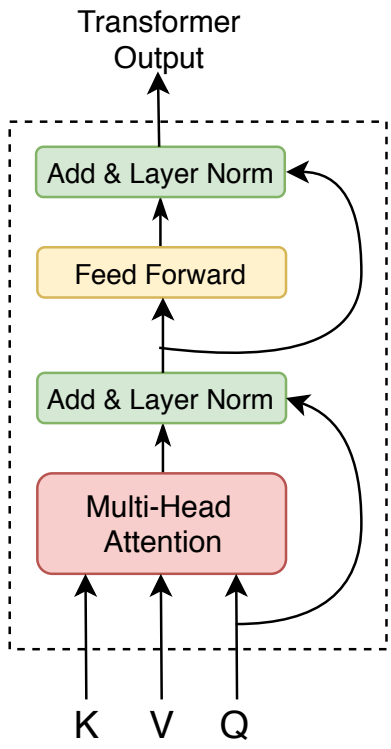

Figure 6.4: Transformer Block.

Figure 6.4 shows a typical Transformer block. A feed forward layer is stacked on top of the Multi-Head Attention module and after each module a residual connection[28] is added followed by layer normalization[29].

6.2.4 Positional Encoding

Since there is no sequential feed of information to the model, it will have no information about the sequence or relative positions of the frames or symbols in the data. To overcome this, the input embedding that is fed to the model is added with a positional encoding. To generate the positional encodings, two sinusoidal functions of different frequencies are used. The equations for the positional encoding are given below.

$$PE(pos, 2i) = sin(\frac{pos}{10000^{\frac{2i}{d}}}) \tag{6.5}$$

$$PE(pos, 2i+1) = cos(\frac{pos}{10000^{\frac{2i}{d}}}) \tag{6.6}$$

Here pos is the position of a frame in the sequence, i corresponds to the i^{th} element in the embedding and d is the size of the embedding. The dimension of the positional encodings will be same as that of the input embeddings so that they can be added together. The positional encoding is hypothesized to help the model in learning to attend with relative positional information.

6.2.5 Model Architecture

Similar to most of the architectures for Seq2Seq models, the Transformers architecture is also made up of an encoder to encode the input sequence and decoder to decode the output sequence from the output of the encoder. Fig. 6.5 shows the transformer model architecture. The left half of the architecture is the encoder, and the right half is the decoder.

Encoder

The input to the encoder is created by adding the frame features with the positional encoding. The input is stacked into a matrix and is given as query, key, and value to the transformer block. This enables the encoder to perform self-attention on the input frames. By the mechanism of self-attention, each frame can attend to every other frame, including itself to gain a better perspective of the video and to generate better representations by including relevant information about every frame. N such transformer blocks are stacked on top of one another to enable multiple levels of self-attention to generate

a deeper representation of the input frames. Dropout regularisation is used on the feed-forward layers to avoid overfitting. The final output of the encoder is called the encoder memory and is of the same dimension as that of the input of the encoder.

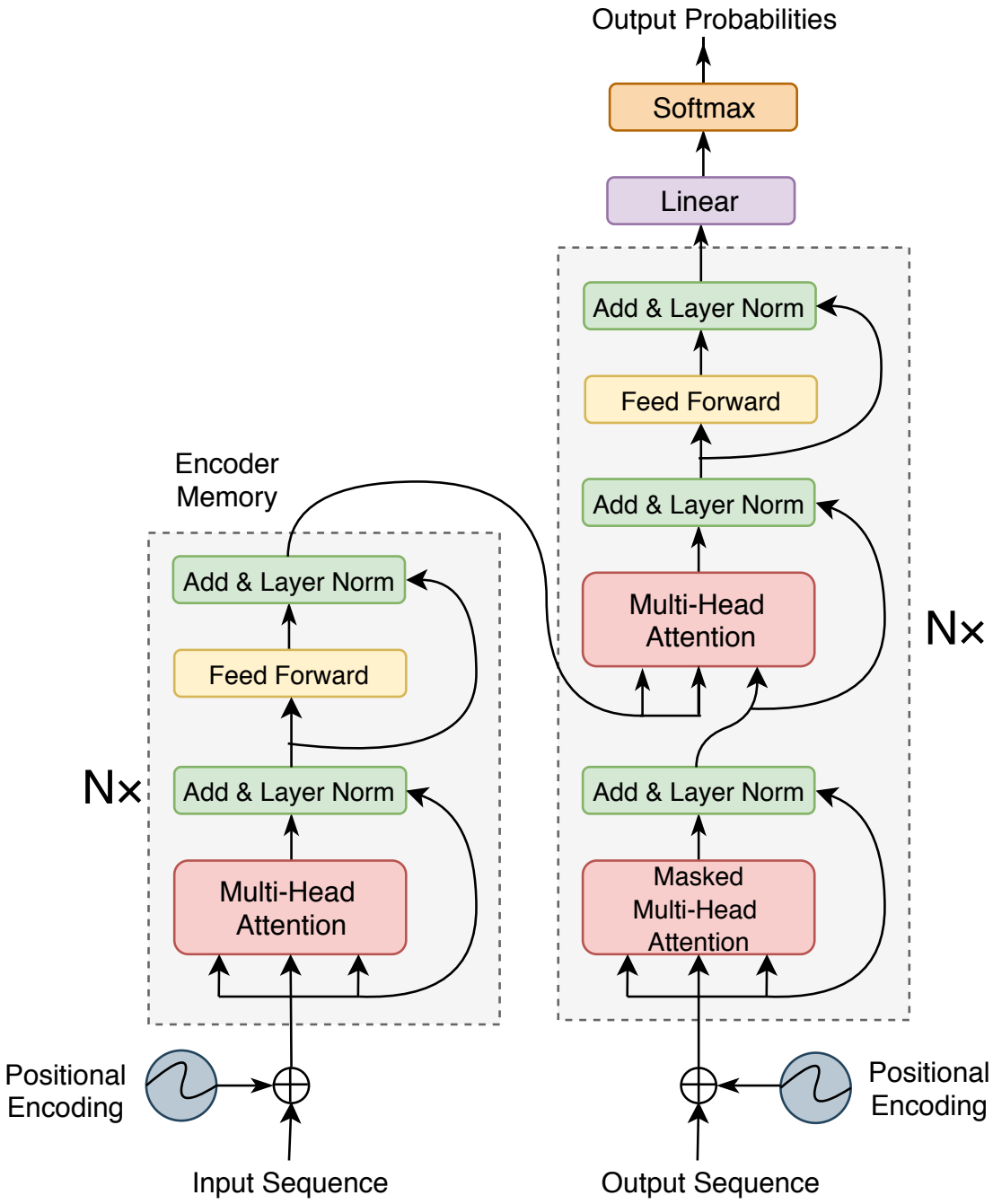

Figure 6.5: Transformers Model Architecture.

Decoder

The input to the decoder is embeddings of the ground truth caption symbols with a <START> symbol in the front. Similar to the encoder inputs, the decoder inputs are

summed with positional encodings. The decoder also has a self-attention module, but it is a restricted form of self-attention, i.e., the input embeddings are not allowed to attend to the embeddings of future symbols. A symbol at timestep t will only be able to attend to symbols from timestep 0 to t. This restriction is imposed owing to the fact that during inference, the symbol at timestep t would be generated before the symbol at timestep $t+1$. Therefore any reference to the future symbols is restricted in the decoder. This restricted self-attention is simply achieved by applying a mask on the unnormalized attention weights in Eq. 6.1. The new equation for restricted self-attention is as follows.

$$RSA(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}} \odot M)V \tag{6.7}$$

Here M is a lower triangular matrix of unity of size n_q(number of queries) and $\odot$ represents element-wise multiplication.

After self-attention, the next attention layer is the one that connects the encoder memory to the decoder. In this attention layer, the output of the decoder's restricted self-attention becomes the queries, and the encoder memory becomes the keys and values. This helps the decoder to attend to the encoder memory. The output of this attention block is then fed to a feed-forward layer. Again N such transformer blocks are stacked on top of one another to enable multiple levels of self-attention and cross-over attentions to generate deeper representations. Dropout regularisation is used on the feed-forward layers to avoid overfitting. A softmax function is applied to the final output of the decoder to get the output probabilities. The targets for the decoder outputs are the one-hot encodings of the ground truth caption symbols without the <START> symbol.

6.3 Experiments

6.3.1 Training

The dataset preparation is identical to the methods discussed in the previous chapters. The number of transformer blocks, N was set to 6, and the number of attention blocks in a multi-head attention module, h was set to 8. Except for the learning rate decay, all the other hyperparameters were unchanged. The initial learning rate was set to 10^{-4}

and the minibatch size was set to 128. The equation for learning rate decay discussed in [13] is as follows.

$$lr = d^{-0.5}.min(step\_num^{-0.5}, step\_num * warmup\_step^{-1.5}) \tag{6.8}$$

Here d is the dimension of key or value vector. This decay method works by linearly increasing the learning rate for the first $warmup\_step$ warmup steps and then decreasing it proportionally to the inverse square root of the $step\_num$. This learning rate scheme is called the Noam learning rate scheme and is depicted in Fig. 6.6. The $warmup\_step$ was set to 4000 and the dropout rate was set to 0.3.

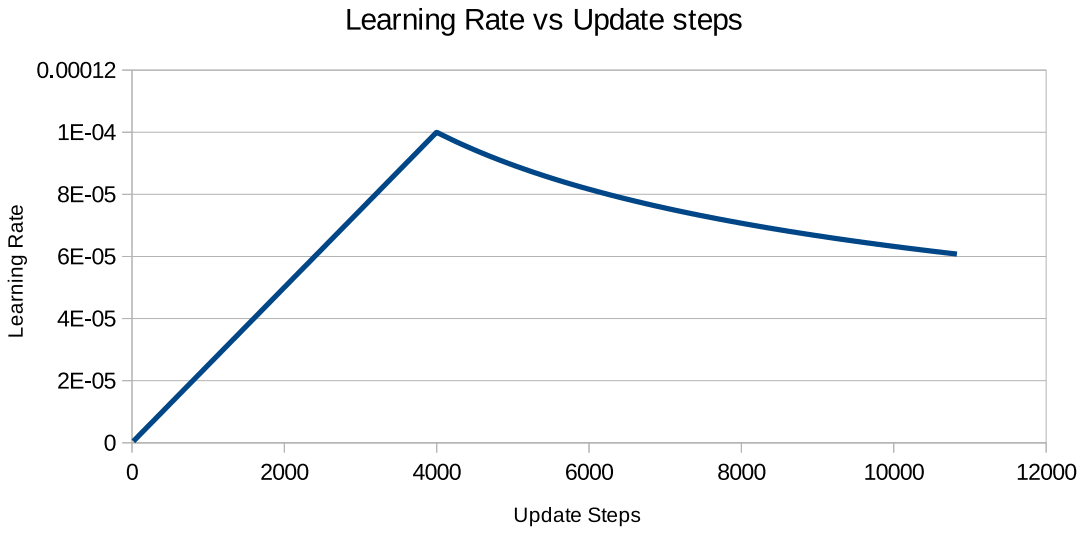

Figure 6.6: Noam Learning rate scheme.

6.3.2 Inference

Although the transformers could be trained without any sequential feed of data, the inference couldn't be done that way. This is because the probability distribution for output symbol at timestep t is conditioned on all the symbols from timestep 0 to $t-1$. So to generate the probability distribution for output symbol at timestep $t+1$, the symbol at t must be clearly defined. Therefore the inference in transformers could only be done in an auto-regressive fashion.

In inference mode, the input frame features are fed to the encoder and the encoder

memory is captured. The input to the decoder is filled with zeros except for the first row corresponding to the first input symbol, which is filled with the embedding of the <START> symbol. In the first forward pass, the output probability distribution for the first output symbol is generated and the rest of the output is ignored. With this distribution, we could find the first output symbol with an argmax function. The second row of the input to the decoder is filled with the embedding of the first output symbol and the rest of the rows are left unaltered. This input goes for a second forward pass and the rest of the symbols are decoded similarly until the <END> symbol is reached.

6.4 Results

The results of video captioning model with transformers on the MSVD dataset compared against LSTM+LM(Chapter 5) is given in Table 6.1.

Table 6.1: Results of Transformer compared with LSTM+LM on MSVD dataset

Model	**B@1**	**B@2**	**B@3**	**B@4**	**METEOR**	**CIDEr**
LSTM+LM	**81.8**	**70.7**	**61.3**	**51.7**	**33.5**	**77.0**
Transformer	80.9	69.1	60.4	48.6	32.0	73.2

Although the transformer framework outperforms state-of-the-art methods on machine translation tasks, it could be seen from Table 6.1 that the Transformer framework did not perform any better than the LSTM+LM framework on video captioning task. This performance drop suggests that the transformer framework on its own might not be enough to model the video captioning data efficiently. It could also be hypothesized that there is a need for recurrent layers to model the video captioning data effectively.

6.5 Summary

In this chapter, we discussed about the transformer framework which was primarily developed for machine translation tasks. This framework uses the attention mechanism alone to model sequence to sequence data without the need for any recurrent layers. We applied this framework for our video captioning task and found that it did not perform any better than our previously proposed approach. In the next chapter, a hybrid

framework that integrates Transformer and LSTM is proposed, and experiments are performed on this framework.

CHAPTER 7

Video Captioning with Attention-based LSTM over Transformer Memory

7.1 Motivation

Although the vanilla transformer framework did not produce superior results than LSTM+LM, the transformer framework had good potential for generating better representations for the frame from its features by correlating other frame features. Since these representations are generated by attending to all the frames, these representations could have a bi-directional view of the video. Therefore this could be a replacement for the encoder in any LSTM based Seq2Seq architecture.

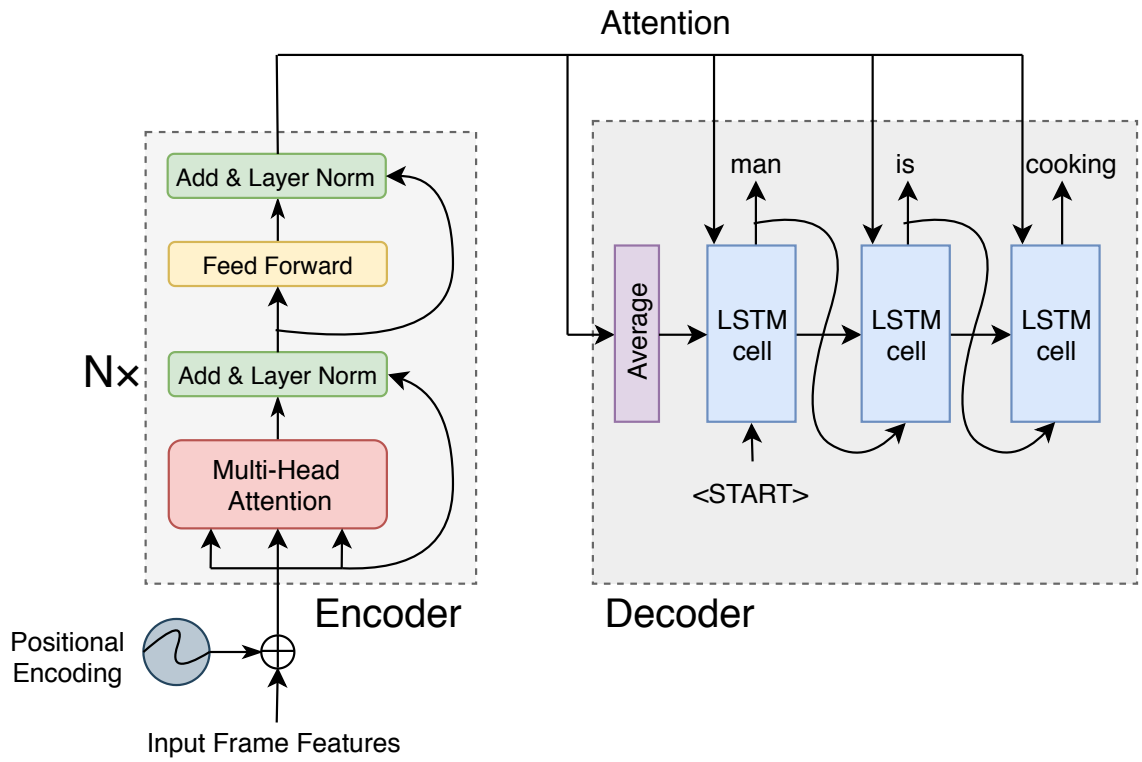

Figure 7.1: LSTM over Transformer Memory.

7.2 Proposed Approach

The transformer encoder memory is a set of M representations corresponding to the M input frames to the encoder. An attention-based LSTM could be used as the decoder for our model by attending to the encoder memory. Further, the initial state of the decoder LSTM could be initialized with the average of the encoder memory. This model is thus end-to-end trainable and can be trained with the sequential cross-entropy loss from the decoder. Fig. 7.1 illustrates the proposed framework. To further enhance the performance of the model, the language model based encoder target loss discussed in chapter 5 could be applied here. The loss functions are now similar to LSTM+LM(chapter 5) and are as follows

$$Loss_1 = -\Sigma_{t=0}^{N_d}(log(P(d_t|d_1, d_2 \ldots, d_{t-1}))) \tag{7.1}$$

$$Loss_2 = \|avg(x) - v\|_{L2} \tag{7.2}$$

$$Loss_{total} = \lambda Loss_1 + (1 - \lambda)Loss_2 \tag{7.3}$$

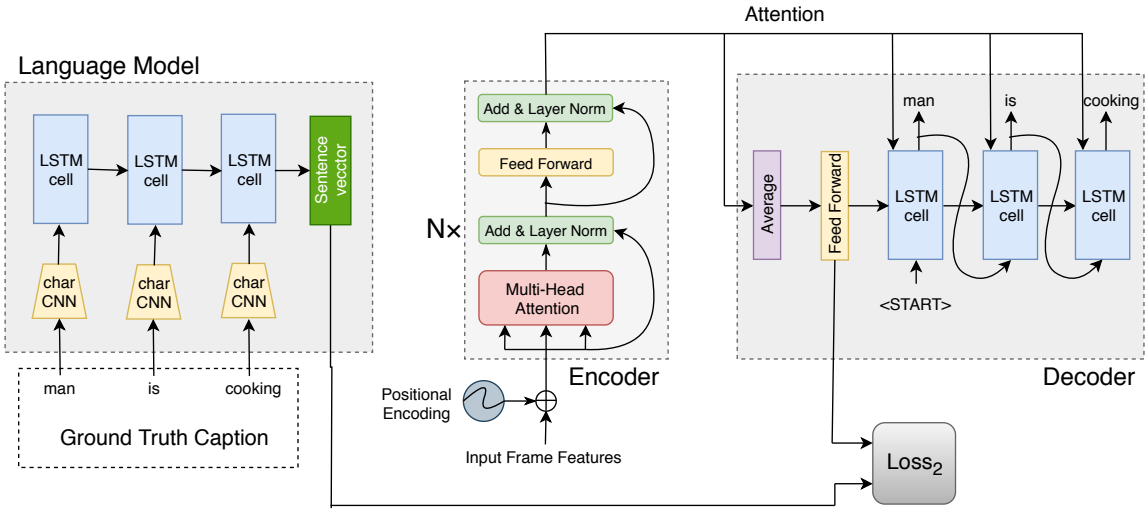

Figure 7.2: LSTM over Transformer Memory with Language-Model based loss.

Again here d_i denotes the i^{th} word of the sentence and N_d is the number of words in the caption. x is the encoder memory from the encoder, v is the sentence vector, λ is a hyper-parameter which acts as a trade-off between the two losses.Fig. 7.2 illustrates

the same framework with language model based loss functions.

7.3 Training

For the encoder, N was set to 3 and h was set to 8. For the decoder the LSTM unit size was set to 1024 and the embedding size was set to 512. The dropout probability was set to 0.2. Adam optimizer with a initial learning rate of 10^{-4} with Noam decay scheme(discussed in Section 6.3.1) was used. λ was set to 0.999 when using language-model based loss. The transformer module was prone to vanishing gradient issues for smaller values of λ. With a minibatch size of 256 this model was trained for 200 epochs with an early stopping patience of 30 epochs. For inference, beam search decoder with a beam size of 9 was used.

7.4 Results

The results of the Transformer and LSTM hybrid model on MSVD dataset are compared against other models in Table 7.1

Table 7.1: Results of Transformer and LSTM hybrid model compared against other techniques on MSVD dataset

Model	**B@1**	**B@2**	**B@3**	**B@4**	**METEOR**	**CIDEr**
LSTM+LM (Chapter 5)	81.8	70.7	61.3	51.7	33.5	**77.0**
Vanilla Transformer (Chapter 6)	80.9	69.1	60.4	48.6	32.0	73.2
LSTM over Transformer Memory	**82.7**	72.0	63.0	52.9	**33.8**	72.6
LSTM over Transformer Memory with Language-Model based loss	**82.7**	**72.5**	**63.4**	**53.3**	33.5	71.1

It could be seen from Table 7.1 that the proposed approach outperforms all the previously discussed methods. It is also clear that augmenting the objective function of the model with a Language-Model based loss has helped to improve the performance of the system significantly in almost all the metrics.

7.5 Summary

In this chapter, a hybrid framework that integrates Transformer and LSTM is proposed, and experiments are performed on this framework. This framework uses the Transformer module as the encoder and LSTM as the decoder. This framework has significantly outperformed all the previously discussed approaches. It was also seen that the model's performance further improved by augmenting the objective function of the model with a Language-Model based loss.

CHAPTER 8

Analysis

8.1 Score comparison with other popular Methods

The scores of LSTM over Transformer Memory model is compared against popular models on the MSVD dataset and the MSR-VTT dataset in Table 8.1 and Table 8.2 respectively.

Table 8.1: Results of LSTM over Transformer Memory model compared against other popular techniques on MSVD dataset

Model	**B@1**	**B@2**	**B@3**	**B@4**	**METEOR**	**CIDEr**
LSTM-TSA [7]	82.8	72.0	62.8	52.8	33.5	**74.0**
hLSTMat (Inception-v3)[6]	82.7	72.0	62.5	51.9	33.5	73.8
hLSTMat (ResNet-152)	**82.9**	72.2	63.0	53.0	**33.6**	73.8
LSTM over Transformer Memory with Language-Model based loss (Chapter 7) (Inception-v3)	82.7	**72.5**	**63.4**	**53.3**	33.5	71.1

Table 8.2: Results of LSTM over Transformer Memory model compared against other popular techniques on MSR-VTT dataset

Model	**B@4**	**METEOR**	**CIDEr**
hLSTMat (ResNet-152)	38.3	26.3	-
SA-LSTM [30]	38.7	**26.9**	**45.9**
LSTM over Transformer Memory with Language-Model based loss (Chapter 7) (Inception-v3)	**39.0**	25.3	39.5

It could be seen from the Table 8.1 and Table 8.2 that the proposed approach outperforms popular approaches on both the datasets. Do note that no ensemble approaches were considered for the comparison.

8.2 Analysis of Generated Captions

Some of the captions generated from the LSTM+LM model and proposed model(transLSTM+LM) (Chapter 7) are compared against the ground truth captions in Figure 8.1.

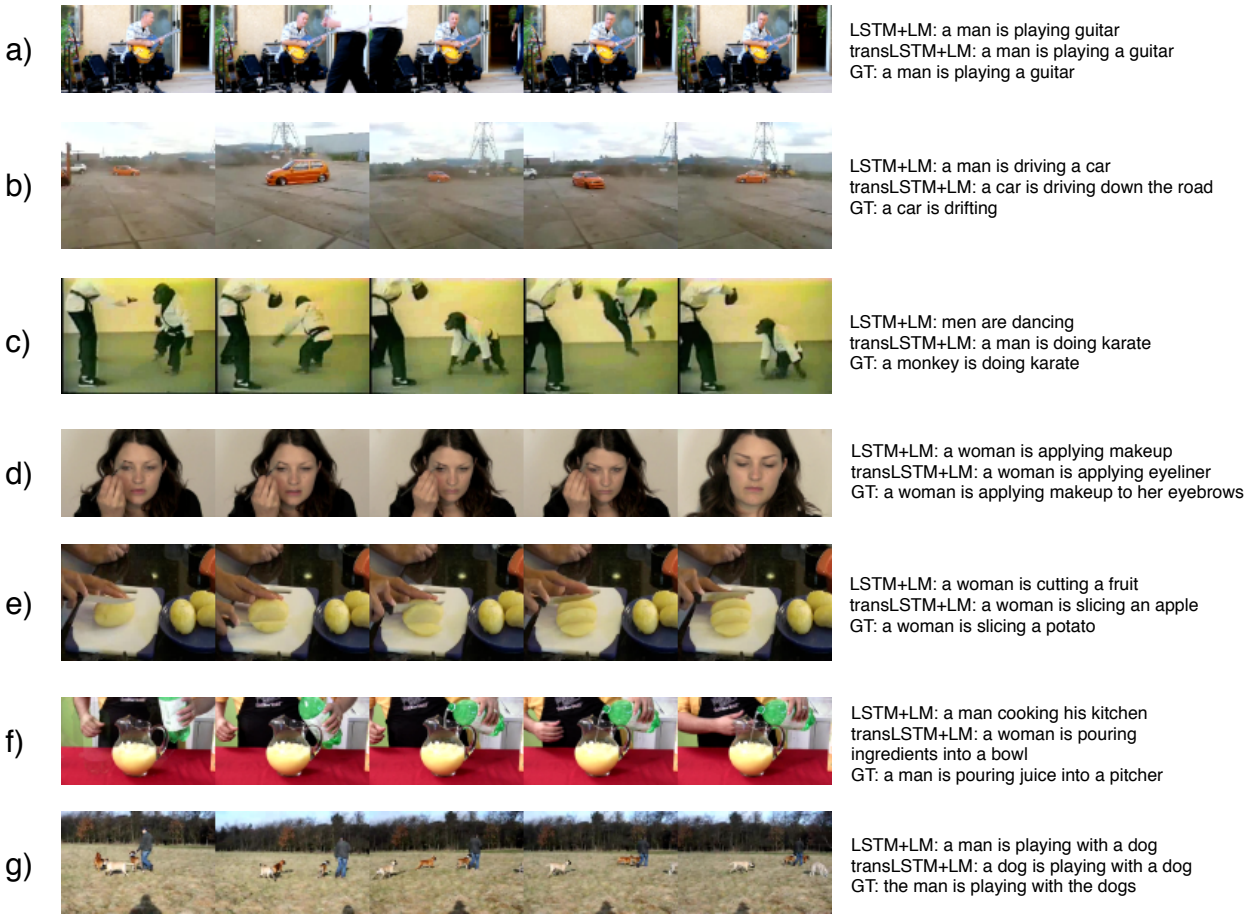

Figure 8.1: Captions from LSTM+LM model, transLSTM+LM model and ground truth(GT)

It could be seen that the captions generated by both models have varying degrees of correctness. For video (a), the captions generated by both the model are very identical to the ground truth. This is not the case for video (b); the caption from LSTM+LM model takes *"man"* as the subject and describes the video; whereas the transLSTM+LM model takes *"car"* as the subject and describes the same video. Although both the captions describe the video in a different sense, they are both quite valid. The same is the case with video (d) and video (g). For video (d), LSTM+LM model describes that the woman is *applying make-up* whereas transLSTM+LM is closer to the ground truth by mentioning that she is *applying an eyeliner*. For video (c), the LSTM+LM model describes that *men are dancing*, but it could be seen that a man and a monkey

are doing karate; the transLSTM+LM model is again little accurate by identifying the action correctly but fails to recognize the monkey in the video. The transLSTM+LM model for video (f) is very accurate in describing the video whereas the LSTM+LM caption has failed to do so.

It could be seen from the sample captions from Figure 8.1 that the transLSTM+LM model has performed better than the LSTM+LM captions and the scores from Table 7.1 confirms the same.

CHAPTER 9

Summary and Conclusion

Various approaches to video captioning were explored, and their performances were compared. Initially, the aLSTMs approach introduced in [2] was implemented, and experiments were performed on this approach. Studies were done to incorporate the knowledge learnt from a pre-trained Language Model into the video captioning system. To achieve this, the objective function of the model was augmented with a loss derived from the Language Model. This augmentation was inspired by aLSTMs approach and experiments(Table 5.2) with this approach(LSTM+LM) showed a significant performance boost. The Transformer framework introduced in [13] was tried for the video captioning task and it was seen that the performance was not nearly as good as the LSTM+LM approach. A novel approach was proposed that integrates LSTM and Transformer and proved to perform better than LSTM+LM in terms of evaluation metrics. It was again evident from the experiments(Table 7.1) that the model trained with the Language-Model based augmented objective function is superior to the model trained without augmented objective function. The scores of this proposed approach with Language-Model based loss achieved near state-of-the-art performance on both the described datasets. The scores could be further improved by utilizing an ensemble method to combine multiple models. Video captioning is a fascinating problem in deep learning and has a long way to go.

REFERENCES

[1] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the limits of language modeling," *arXiv preprint arXiv:1602.02410*, 2016.

[2] L. Gao, Z. Guo, H. Zhang, X. Xu, and H. T. Shen, "Video captioning with attention-based lstm and semantic consistency," *IEEE Transactions on Multimedia*, vol. 19, no. 9, pp. 2045–2055, Sept 2017.

[3] L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville, "Describing videos by exploiting temporal structure," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ser. ICCV '15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 4507–4515. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2015.512

[4] H. Yu, J. Wang, Z. Huang, Y. Yang, and W. Xu, "Video paragraph captioning using hierarchical recurrent neural networks," *CoRR*, vol. abs/1510.07712, 2015. [Online]. Available: http://arxiv.org/abs/1510.07712

[5] P. Pan, Z. Xu, Y. Yang, F. Wu, and Y. Zhuang, "Hierarchical recurrent neural encoder for video representation with application to captioning," *CoRR*, vol. abs/1511.03476, 2015. [Online]. Available: http://arxiv.org/abs/1511.03476

[6] J. Song, Z. Guo, L. Gao, W. Liu, D. Zhang, and H. T. Shen, "Hierarchical LSTM with adjusted temporal attention for video captioning," *CoRR*, vol. abs/1706.01231, 2017. [Online]. Available: http://arxiv.org/abs/1706.01231

[7] Y. Pan, T. Yao, H. Li, and T. Mei, "Video captioning with transferred semantic attributes," *CoRR*, vol. abs/1611.07675, 2016. [Online]. Available: http://arxiv.org/abs/1611.07675

[8] D. L. 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 - Volume 1*, ser. HLT '11. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 190–200. [Online]. Available: http://dl.acm.org/citation.cfm?id=2002472.2002497

[9] Y. Pan, T. Mei, T. Yao, H. Li, and Y. Rui, "Jointly modeling embedding and translation to bridge video and language," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[10] J. Xu, T. Mei, T. Yao, and Y. Rui, "Msr-vtt: A large video description dataset for bridging video and language," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[12] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. [Online]. Available: https://www.aclweb.org/anthology/D14-1179

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 5998–6008. [Online]. Available: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

[14] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training Recurrent Neural Networks," *ArXiv e-prints*, Nov. 2012.

[15] J. Dong, X. Li, and C. G. M. Snoek, "Predicting Visual Features from Text for Image and Video Caption Retrieval," *ArXiv e-prints*, Sep. 2017.

[16] C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, and P. Koehn, "One billion word benchmark for measuring progress in statistical language modeling," *CoRR*, vol. abs/1312.3005, 2013. [Online]. Available: http://arxiv.org/abs/1312.3005

[17] X. Zhang, J. J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," *CoRR*, vol. abs/1509.01626, 2015. [Online]. Available: http://arxiv.org/abs/1509.01626

[18] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," *ArXiv e-prints*, Sep. 2014.

[19] R. F. L. T. D. Tran, L. Bourdev and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 4489–4497, Dec 2015.

[20] K. Cho, A. C. Courville, and Y. Bengio, "Describing multimedia content using attention-based encoder-decoder networks," *CoRR*, vol. abs/1507.01053, 2015. [Online]. Available: http://arxiv.org/abs/1507.01053

[21] C. Liu, J. Mao, F. Sha, and A. Yuille, "Attention Correctness in Neural Image Captioning," *ArXiv e-prints*, May 2016.

[22] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang, "Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering," *ArXiv e-prints*, Jul. 2017.

[23] E. Loper and S. Bird, "Nltk: The natural language toolkit," in *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ser. ETMTNLP '02. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 63–70. [Online]. Available: https://doi.org/10.3115/1118108.1118117

[24] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *CoRR*, vol. abs/1512.00567, 2015. [Online]. Available: http://arxiv.org/abs/1512.00567

[25] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, Dec 2015. [Online]. Available: https://doi.org/10.1007/s11263-015-0816-y

[26] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[29] J. Lei Ba, J. R. Kiros, and G. E. Hinton, "Layer Normalization," *arXiv e-prints*, p. arXiv:1607.06450, Jul 2016.

[30] J. Dong, X. Li, W. Lan, Y. Huo, and C. G. Snoek, "Early embedding and late reranking for video captioning," in *Proceedings of the 24th ACM International Conference on Multimedia*, ser. MM '16. New York, NY, USA: ACM, 2016, pp. 1082–1086. [Online]. Available: http://doi.acm.org/10.1145/2964284.2984064