4-2017

Automatic Video Captioning using Deep Neural Network

Thang Huy Nguyen
thn2079@rit.edu

Follow this and additional works at: http://scholarworks.rit.edu/theses

Recommended Citation

This Thesis is brought to you for free and open access by the Thesis/Dissertation Collections at RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact rit-scholarworks@rit.edu.
Automatic Video Captioning using Deep Neural Network

Master’s Thesis

Thang Huy Nguyen

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Engineering
Supervised by
Dr. Raymond Ptucha
Department of Computer Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, NY
04/2017

Approved By:

Date:
Dr. Raymond Ptucha
Primary Advisor – R.I.T. Dept. of Computer Engineering

Date:
Dr. Nathan Cahill
Secondary Advisor – R.I.T. School of Mathematical Sciences

Date:
Dr. Dhireesha Kudithipudi
Secondary Advisor – R.I.T. Dept. of Computer Engineering
Acknowledgements

I would like to thank professor Ptucha, my amazing mentor, who has always motivated, encouraged and given me valuable advice throughout the research. I would like to thank my colleague Shagan Sah for his feedback and his help to this work. Finally, I would like to thank Machine Intelligence Lab of Rochester Institute of Technology for providing me necessary resources to complete this project.
I dedicate this thesis to my father and my mother for their unconditional love and support.
Abstract

Video understanding has become increasingly important as surveillance, social, and informational videos weave themselves into our everyday lives. Video captioning offers a simple way to summarize, index, and search the data. Most video captioning models utilize a video encoder and captioning decoder framework. Hierarchical encoders can abstractly capture clip level temporal features to represent a video, but the clips are at fixed time steps. This thesis research introduces two models: a hierarchical model with steered captioning, and a Multi-stream Hierarchical Boundary model. The steered captioning model is the first attention model to smartly guide an attention model to appropriate locations in a video by using visual attributes. The Multi-stream Hierarchical Boundary model combines a fixed hierarchy recurrent architecture with a soft hierarchy layer by using intrinsic feature boundary cuts within a video to define clips. This thesis also introduces a novel parametric Gaussian attention which removes the restriction of soft attention techniques which require fixed length video streams. By carefully incorporating Gaussian attention in designated layers, the proposed models demonstrate state-of-the-art video captioning results on recent datasets.
Table of Contents

ABSTRACT ........................................................................................................................................... 4

TABLE OF CONTENTS ....................................................................................................................... 5

LIST OF FIGURES ............................................................................................................................ 7

LIST OF TABLES ............................................................................................................................... 9

CHAPTER 1 INTRODUCTION .............................................................................................................. 10
  1.1 NOVEL CONTRIBUTIONS ........................................................................................................... 11
  1.2 MOTIVATION FROM PREVIOUS WORK ................................................................................ 12

CHAPTER 2 CONVOLUTIONAL NEURAL NETWORK .......................................................................... 14
  2.1 CONVOLUTIONAL LAYER ............................................................................................................ 15
  2.2 POOLING LAYER ....................................................................................................................... 17
  2.3 ACTIVATION FUNCTIONS ......................................................................................................... 18
  2.4 COMMON CNN ARCHITECTURES ............................................................................................ 19

CHAPTER 3 RECURRENT NEURAL NETWORKS .................................................................................. 20
  3.1 VANILLA RECURRENT NEURAL NETWORK .............................................................................. 21
  3.1.1 BACK PROPAGATION THROUGH TIME .............................................................................. 23
  3.1.2 VANISHING AND EXPLODING GRADIENTS PROBLEM .................................................. 25
  3.2 LONG-SHORT TEM MEMORY NETWORK .............................................................................. 27
  3.3 RECURRENT HIGHWAY NETWORK ......................................................................................... 29

CHAPTER 4 VIDEO CAPTIONING ..................................................................................................... 31
  4.1 ENCODER-DECODER MODELS ............................................................................................... 34
  4.2 SOFT ATTENTION ...................................................................................................................... 37

CHAPTER 5 METHODOLOGY ............................................................................................................. 39
  5.1 GAUSSIAN ATTENTION ............................................................................................................ 40
  5.2 ATTENTION STEERING ............................................................................................................. 43
  5.3 VIDEO2VEC REPRESENTATION ............................................................................................... 45
  5.4 SHOT BOUNDARY DETECTION ................................................................................................ 47
  5.5 MULTI-STREAM HIERARCHICAL BOUNDARY MODEL .......................................................... 48
  5.6 GAUSSIAN STEERING MODEL ................................................................................................ 50
  5.7 LOSS FUNCTION ....................................................................................................................... 51

CHAPTER 6 DATASET AND TRAINING DETAILS .............................................................................. 53
  6.1 DATASET ................................................................................................................................. 53
<table>
<thead>
<tr>
<th>FIGURE</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High-level visualization of a convolutional neural network</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>An example of input volume is computed in the first convolutional layer</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>Example filters learned by AlexNet</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>Example of a pooling layer</td>
<td>17</td>
</tr>
<tr>
<td>5</td>
<td>Illustration of ReLU function</td>
<td>18</td>
</tr>
<tr>
<td>6</td>
<td>Vanilla RNN</td>
<td>21</td>
</tr>
<tr>
<td>7</td>
<td>An unfolding in time recurrent neural network</td>
<td>22</td>
</tr>
<tr>
<td>8</td>
<td>A simple RNN</td>
<td>23</td>
</tr>
<tr>
<td>9</td>
<td>Inside LSTM module</td>
<td>27</td>
</tr>
<tr>
<td>10</td>
<td>Schematic of an RHN layer inside the recurrent loop</td>
<td>29</td>
</tr>
<tr>
<td>11</td>
<td>Mean-pool architecture</td>
<td>34</td>
</tr>
<tr>
<td>12</td>
<td>S2VT architecture</td>
<td>35</td>
</tr>
<tr>
<td>13</td>
<td>Stacked LSTM video encoder</td>
<td>35</td>
</tr>
<tr>
<td>14</td>
<td>HRNE model</td>
<td>36</td>
</tr>
<tr>
<td>15</td>
<td>High-level visualization of soft attention to video description generation</td>
<td>37</td>
</tr>
<tr>
<td>16</td>
<td>Illustration of the parameterized Gaussian attention model for steering the</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>temporal alignment between the video and word sequence</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Attention steering using normalized temporal feature relevance</td>
<td>44</td>
</tr>
<tr>
<td>18</td>
<td>Overview of the multi-stream hierarchical boundary model for video captioning</td>
<td>49</td>
</tr>
</tbody>
</table>
List of Tables

Table 1. Video-sentence pair dataset statistics. .............................................................. 53
Table 2. A summary of the evaluation metrics. .............................................................. 57
Table 3. BLEU score computation example ................................................................... 59
Table 4. MSVD caption evaluation results with MHB model. ......................................... 65
Table 5. MSR-VTT results on held-out test set. ............................................................... 66
Table 6 Comparison of captioning results on MVAD dataset. ........................................ 67
Table 7 MSVD caption evaluation results on the held-out test set. ................................. 69
Table 8 HGA results on the held out MSR-VTT test set. .............................................. 70
Table 9 MSR-VTT results on the held-out test set. We compare with recent entries in the
   MSR Video to Language Challenge. ........................................................................... 72
Table 10 Comparing Gaussian attention at different layers for MSR-VTT test set. Adding
   GA show clear improvement over SA and attention is most important at the word
   generation layer........................................................................................................... 73
Table 11 Performance evaluation with number of Gaussian filters 532 for attention on the
   MSVD test set. ............................................................................................................ 76
Table 12 Illustration of Gaussian shape correspond with word in the caption sentence. . 79
Table 13 Illustration of Gaussian shape correspond with word in the caption sentence. . 81
Chapter 1  Introduction

Deep learning has recently revolutionized computer vision in many applications. By learning deep features and representations, a machine can match or even beat human performance in various tasks such as image classification, object recognition, and video segmentation. However, tasks like image and video captioning still remain a challenge and have captured an immense attention from literature in the past two years. The nature of a video stream with high temporal dependencies, multiple scenes in a complex video and diverse kinds of objects and actions make captioning very difficult. Yet, despite this, many architectures and methods have been proposed which substantially push forward research in video description. Building upon those successes, this thesis work develops robust captioning frameworks which can automatically generate captions for both simple and complex videos.
1.1 Novel Contributions

- Gaussian parametric attention: Soft attention models have an intrinsic limitation that all input buffers need to be of the same duration. This is because the attention vector is associated with a learnable, but fixed dimension weight matrix. For videos, this requires cropping longer videos or padding shorter videos. The proposed parametric Gaussian attention model removes this limitation by applying a continuous, rather than discrete weight distribution.

- Temporal steering. Existing attention models are guided by temporal features of the training data. For example, when the phrase winning goal occurs, the attention might jump towards the end of video. The introduced temporal attention steering mechanism uses frame level visual concepts to guide attention based on current video properties (detection of objects, activities, etc.) and not on training data trends.

- Introduction of an intelligent hierarchy for encoding a video. This hierarchy is based on intrinsic video boundaries. We refer to this as a Multi-stream Hierarchical Boundary (MHB) model, and further adapt Recurrent Highway Networks for the captioning task.

- We train fully end-to-end models that learns from video features to determine the temporal bounds of video clips to generate text descriptions of an entire video and demonstrate state-of-the-art captioning results on recent video datasets.
1.2 Motivation from previous work

Image and video understanding has gained a lot of attention in deep learning research recently. Image classification [1-4], object detection [5, 6], semantic segmentation [7-9], image captioning [10-12], and localized image description tasks [13] have witnessed tremendous progress in the last few years.

Automatically describing videos with natural language text enables more efficient search and retrieval, can aid visual understanding in the medical, security, and military applications, and can even be used to describe pictorial content to the visually impaired. Attention models learn to select the most relevant segments that associate with the text.

Spatial [14], temporal [15] [16, 17] and attribute [18] based attention models strive to localize objects in image frames, actions in videos or attend to specific word attributes. The ability to understand how well temporal attention works on video is limited given that most of the popular datasets are comprised of short videos. For example, the average video duration of the MSVD [19] YouTube clips is 10.2 seconds and the average duration of M-VAD [20] movie descriptions clips is 5.8 seconds.

Current methods for generating attention weights are determined by hard training statistics, not by visual concepts. For example, if training videos often end with humans falls, the model may learn to associate end of video as it predicts the words “falls down”.
If a video and corresponding caption are “woman scares man skateboarding and he falls down”, the attention model would perform as expected. But if the caption is “man falls down and then does great skateboard trick”, the trained attention parameters would perform poorly as the model would seek attention over the wrong temporal location in the video.

To steer the attention mechanism to the proper region in the video, our model extracts frame-wise temporal concepts across the length of the video. This enables the model to correlate specific words such as woman, man, and skateboarding, with region-specific locations across the video.

To steer attention towards appropriate spatio-temporal locations in a video, attention models weigh some regions high and others low. The model learns how to linearly combine multiple regions of a video by applying a convex combination of each. As weights are learned parameters, and parameters need to be fixed at train time, attention models are constrained such that all samples have equivalent number of regions. To enable the attention mechanism to be independent of video duration, we present parametric Gaussian attention models which learn a continuous function and samples this function at each discrete region.
Chapter 2  Convolutional Neural Network

Convolution neural networks (CNNs) are a special type of neural network which focus on data that has grid-like topology. CNNs have been extremely successful in practical applications and are adapted in many architectures such as: image recognition, object detection, object segmentation [1, 3, 4]. By combining multiple blocks and using small filter sizes, CNNs can learn an in-depth representation of input, thus make it surpass all other traditional methods in image-related tasks. To fully understand how CNNs work, we need to investigate certain operations that every Convolutional Neural Network has: convolution, pooling, and activation functions.

*Figure 1  High-level visualization of a convolutional neural network[21].*
2.1 Convolutional layer

The convolutional layer is core building block of a Convolution Neural Network (CNN). In this layer, most of the computational operations are conducted. Each convolutional layer using a numerous number of small sized sliding windows (also called as kernels, filters or feature detector), multiply with the input element-wise to get the full convolution. For example, in Figure 2, the input is an RGB CIFAR-10 image with the size of 32×32×3. If this image were convolved with 16 filters with size of 5×5×3 and stride of 1 at the first layer, the output of 28×28×16. To make the output dimension the same as the input dimension, the images are often padded with zeroes. The depth, stride and number of zero padding are three hyperparameters in a convolutional layer that control the output volume. The depth is the number of filters we would like to choose, where each filter will learn some features from the input. The stride controls how the filter will slide through the input, the larger of the stride the smaller of the output spatially. Zero-padding is employed in some cases, mostly to preserve the size of input volume.

As [21] suggests, we can calculate the output of a convolutional layer as follow:

- Assume we have input of $W_1 \times H_1 \times D_1$

- Hyperparameters:
  - Number of filters $K$
  - Each filter has spatial extent $F$
  - Stride is $S$,
  - Number of padding $P$

- Output of this convolutional layer is $W_2 \times H_2 \times D_2$, with:
o $W_2 = (W_1 - F + 2P)/S + 1$

o $H_2 = (H_1 - F + 2P)/S + 1$

o $D_2 = K$

Figure 2 An example of input volume (32x32x3 CIFAR-10 image) is computed in the first convolutional layer[21]

Figure 3 shows the visualization of the first convolutional layer in the Alexnet[1] model. There are total 96 filters, each has size of 11x11x3, we can observe that in this layer the model is able to learn filters which have Gabor-like features.

Figure 3 Example filters learned by Alexnet. Each of the 96 filters shown here is of size [11x11x3], and each one is shared by the 55*55 neurons in one depth slice[21]
2.2 Pooling layer

Pooling layers are periodically inserted after each convolutional layer to reduce the dimensionality of the network.

Generally, the pooling layer take:

- Input with size \( W_1 \times H_1 \times D_1 \)
- With hyperparameters:
  - Spatial extent \( F \)
  - Stride \( S \)
- Will produce output \( W_2 \times H_2 \times D_2 \), with:
  - \( W_2 = (W_1 - F)/S + 1 \)
  - \( H_2 = (H_1 - F)/S + 1 \)
  - \( D_2 = D_1 \)

In the example in Figure 4, the input has size of 224\( \times \)224\( \times \)64. After pooling with \( F=2 \), \( S=2 \), the output has volume of size 112\( \times \)112\( \times \)64. The most common technique for pooling is max-pooling [1, 3, 4]. However, [22] suggests that in large scale networks, max-pooling can be substituted with convolutional layer with increased stride without loss in accuracy.
2.3 Activation Functions

To model complex functions, neural networks insert non-linear activation functions between each layer. Many non-linearity activation functions have been proven effective since neural networks were introduced, such as: sigmoid function, tanh function, and Rectified Linear Unit (ReLU) function. The ReLU is the method of choice for most current architectures. The ReLU function computes the function:

\[ f(x) = \max(0, x) \]  \hspace{1cm} (1)

ReLU is very popular since [1] found out that it helped stochastic gradient descent converge much faster (a factor of six) than the tanh function. This can be attributed to its linear, non-saturated form. Moreover, ReLU has a simplistic implementation which does not involve expensive computations like sigmoid/tanh functions.
2.4 Common CNN Architectures

Several CNN architectures have been proposed and shown to be effective in real-world applications:

- LeNet [23]: The first successful convolution neural network by Yann Lecun.
- Alexnet [1]: This is the work popularized the convolutional networks, proposed by Alex Krizhevsky, Ilya Sutskever and Geoff Hinton.
- GoogLeNet [3]: ILSVRC 2014 winner, by introducing an Inception module, this architecture drastically reduces the number of learnable parameters. Interestingly, they replace the fully connecter layer with average pooling at the top of the network.
- VGGnet [2]: This model achieved second place in ILSVRC2014. This architecture has proved that the depth of the network is more important than breadth for a successful model.
- ResNet: Developed by Kaiming He et al.[4], winner of ILSVRC 2015. This is the model which obtains state of the art performance. It introduces skip connections with substantial use of batch normalization. In this thesis, most of our models use ResNet to facilitate our Recurrent Neural Networks layer to achieve the best results.
Chapter 3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are types of neural networks that can process sequential data. While CNNs mostly focus on processing on fixed data at a time with a grid of values such as images, RNNs particularly handle data with temporal dependencies (data input changes over time). In theory, RNNs can handle arbitrary sizes of input and output, enabling the development of flexible neural network models.

Recently RNNs have been used in a wide variety of applications such as speech recognition, language modeling, image captioning, and video descriptions. In this research, we utilize two variants of RNNs, namely LSTM and RHN.
3.1 *Vanilla recurrent neural network*

To explain how RNNs work, first let’s examine a simple Recurrent Neural Network, here namely Vanilla Recurrent Neural Network. The idea of RNNs is to learn dependencies of input sequences. In RNNs, the input to the next step depends on previous computation (Figure 6).

\[ h_t = f(Ux_t + W h_{t-1}) \]  

(2)

with \( f \) is non-linear function like tanh or ReLU.

Then we can calculate output at step \( t \). Assume we need to predict the next word of a sentence. We have to calculate the probabilities for an entire dictionary of words:

\[ o_t = softmax(V s_t) \]  

(3)

Assume we have input \( x_t \) at time \( t \). The hidden state \( h_t \) at \( t \) can be computed as:

\[ h_t = f(U x_t + W h_{t-1}) \]  

Assume we have input \( x_t \) at time \( t \). The hidden state \( h_t \) at \( t \) can be computed as:

\[ h_t = f(U x_t + W h_{t-1}) \]  

(2)

with \( f \) is non-linear function like tanh or ReLU.

Then we can calculate output at step \( t \). Assume we need to predict the next word of a sentence. We have to calculate the probabilities for an entire dictionary of words:

\[ o_t = softmax(V s_t) \]  

(3)
By unfolding the network in Figure 6, we have the unrolled network as illustrated in Figure 7.

Figure 7. An unfolding in time recurrent neural network. [24]
3.1.1 Back propagation through time

Back propagation through time is the main technique used to train an RNN model. Assume we need to train a model for a machine translation application, our hidden state and output are defined as (4) and (5):

\[ s_t = \tanh(Ux_t + Ws_{t-1}) \]  

\[ \hat{y}_t = \text{softmax}(Vs_t) \]  

Our loss function described by cross entropy loss:

\[ E_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t \]  

\[ E(y, \hat{y}) = \sum_t E_t(y_t, \hat{y}_t) = - \sum_t y_t \log \hat{y}_t \]
where $y_t$ is the correct word at time step $t$, $\hat{y}_t$ is our prediction. To calculate gradients, we use chain rule of differentiation. Assume we want to calculate gradients with respect to learning parameters $U$, $V$ and $W$ at loss $E_3$:

\[
\frac{\delta E_3}{\delta V} = \frac{\delta E_3}{\delta \hat{y}_3} \frac{\delta \hat{y}_3}{\delta V} 
\]

\[
= \frac{\delta E_3}{\delta \hat{y}_3} \frac{\delta \hat{y}_3}{\delta z_3} \frac{\delta z_3}{\delta V} 
\]

\[
= (\hat{y}_3 - y_3) \otimes s_3 
\]

\[
\frac{\delta E_3}{\delta W} = \frac{\delta E_3}{\delta \hat{y}_3} \frac{\delta \hat{y}_3}{\delta s_3} \frac{\delta s_3}{\delta W} 
\]

\[
= \sum_{k=0}^{3} \frac{\delta E_3}{\delta \hat{y}_3} \frac{\delta \hat{y}_3}{\delta s_3} \frac{\delta s_3}{\delta s_k} \frac{\delta s_k}{\delta W} 
\]
3.1.2 Vanishing and exploding gradients problem

The vanishing and exploding gradients are a common problem when training RNN models. Due to the propagation of derivatives across several time steps, repeated multiples of values less than one tend towards zero while repeated multiples of values above one explode. RNNs require derivatives to be repeatedly multiplied over times steps, thus being susceptible to vanishing and exploding gradients. As per (8), the notion $\frac{\delta s_3}{\delta s_k}$ is a chain rule, too. We can rewrite (8) as:

$$\frac{\delta E_3}{\delta W} = \sum_{k=0}^{3} \frac{\delta E_3}{\delta y_3} \frac{\delta y_3}{\delta s_3} \left( \prod_{j=k+1}^{3} \frac{\delta s_j}{\delta s_{j-1}} \right) \frac{\delta s_k}{\delta W}$$

(10)

Notice that $\prod_{j=k+1}^{3} \frac{\delta s_j}{\delta s_{j-1}}$ is a product of Jacobian matrices with point-wise derivatives since each element derivative is a vector function with respect to a vector. As proved by Pascanu et al. [25], on certain conditions each of these matrices can have an upper and lower bound of 1. If they have upper bound of 1, the product of those matrices go to 0 exponentially fast with the small values in the matrix and long-range dependencies (normal RNNs architectures have high number of time steps, such as in this research caption decoder has $l=35$). This leads to vanishing gradients and thus the model cannot learn the long-term dependencies. Vice versa, if those Jacobian matrices have a lower bound of 1, we have the exploding gradients problem. However, exploding gradients are easy to find.
out: the loss becomes NaN (Not a Number) and we can efficiently avoid it by clipping gradients to a fixed constant, such as 5.0.
3.2 Long-Short Term Memory Network

To mitigate vanishing and exploding gradients problem as discussed in 3.1.2, Hochreiter et al. [26] proposed the Long Short-Term Memory (LSTM) architecture. The core idea behind LSTMs are cell states and gates. There are three gates in an LSTM which monitor the information flow through the cell: the forget gate, input gate and output gate.

The forget gate controls what information will be ignored from the cell state. The sigmoid activation (10) will output a value between 0 and 1, with 1 is “keep it” and 0 is “forget it”.

We have $f_t$ computed by:

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$  \hspace{1cm} (11)

with $\sigma$ is sigmoid function, $W_f, U_f, b_f$ are learnable weight matrices and bias, $h_{t-1}$ is the previous hidden state of LSTM cell, and $x_t$ is the input vector.

The input gate $i_t$ determines what information will be stored in the cell state. At first, a sigmoid function will decide the values will be updated:
\[ i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i) \]  \hspace{1cm} (12)

\((W_i, U_i, b_i \text{ are learnable weight matrices and bias})\)

A memory cell state \(\bar{C}_t\) at time \(t\) is computed by:

\[ \bar{C}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \]  \hspace{1cm} (13)

\((W_c, U_c, b_c \text{ are learnable weight matrices and bias})\)

The memory cell state \(C_t\) at time \(t\) is updated. The forget gate is multiplied by the forget gate with old state, resulting in the old state with information needed to be forget. Then we add the new candidate value which is scaled by input gate \(i_t \ast \bar{C}_t\):

\[ C_t = i_t \ast \bar{C}_t + f_t \ast C_{t-1} \]  \hspace{1cm} (14)

Equation (14) is the key to alleviating the vanishing gradient problem in RNNs. The derivative from \(C_t\) to \(C_{t-1}\) is the value of the forget gate, and the error at \(C_t\) can pass back to \(C_{t-1}\) without loss with no repeated weight matrix multiplication. This is referred to as Constant Error Carousel [26].

Finally, we decide what we will output to by output gate \(o_t\), then deliver the output hidden state \(h_t\):

\[ o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \]  \hspace{1cm} (15)

\[ h_t = o_t \ast \tanh(C_t) \]  \hspace{1cm} (16)

\((W_o, U_o, b_o \text{ are learnable weight matrices})\)
3.3 Recurrent Highway Network

Vanishing and exploding gradients are not the only problem with the traditional RNN architectures. Training recurrent networks by stacking many layers to create depth is difficult. Intuitively, training a deep recurrent neural network can gain better accuracy but requires huge space and time trade-off. Recently, Zilly et al. [27] introduced Recurrent Highway Networks (RHNs) with the ability to train deeper models with less parameters. A RHN cell is comprised of multiple highway layers, each with two gates- the transform and carry gates. Given an input $x^{[t]}$, $t$ is the current time step, the output of a RHN is described as:

$$s^{[t]}_1 = h^{[t]}_1 \odot t^{[t]}_1 + s^{[t]}_{l-1} \odot c^{[t]}_1$$

(17)
\[ h_t^{[\ell]} = \tanh \left( W_H x_t^{[\ell]} I_{(t=1)} + R_{HI} s_{t-1}^{[\ell]} + b_{HI} \right) \]  

(18)

\[ t_t^{[\ell]} = \tanh \left( W_T x_t^{[\ell]} I_{(t=1)} + R_{TI} s_{t-1}^{[\ell]} + b_{TI} \right) \]  

(19)

\[ c_t^{[\ell]} = \tanh \left( W_C x_t^{[\ell]} I_{(t=1)} + R_{CI} s_{t-1}^{[\ell]} + b_{CI} \right) \]  

(20)

where, \( \ell \) denotes recurrent depth of RHN; \( \odot \) is element-wise function; \( I_{(t)} \) is indicator function; \( W_{H,T,C} \in R^{n \times m} \) and \( R_{(HI, TI, CI)} \in R^{n \times n} \) are transform weight matrices and \( b_{(HI, TI, CI)} \in R^n \) is the bias. It can be observed that RHN conceptually is a variant of LSTM if \( \ell = 1 \), but RHN layers are designed to expand with \( \ell > 1 \), thus enabling complex state transitions which lead to better remembering, forgetting or carrying information. Proved by [27], this design promises to be a better method to train deep recurrent networks while still alleviating the vanishing/exploding gradients problem.

In this research, we incorporate a variant of Recurrent Highway Networks to evaluate its performance against LSTM.
Chapter 4  Video captioning

The success of deep learning in the still image domain has influenced research in the video understanding domain, including video captioning. Early work on video captioning relied on extracting semantic content such as subject, verb, object, and associating it with the visual elements [28-30]. For instance, [29] form a Factor Graph Model to obtain the probability for the semantic content and then use a search based optimization to get the best combination of subject, verb and object to fit in a sentence template. Earlier works were also limited to activity or context specific videos with a small vocabulary of objects and activities. With the availability of large video-sentence pair datasets with rich language information, recent studies [10, 31] have demonstrated use of neural networks to directly model language conditioned on video. Deep neural network architectures for video classification are now prevalent [32, 33].

Initial works that introduced recurrent neural networks for video captioning used a mean pooled feature as the video representation [31]. An alternate approach uses an encoder-decoder [34] framework that first encodes \( f \) frames, one at a time to the first layer of a two layer Long-Short-Term Memory (LSTM), where \( f \) can be of variable length. This latent representation is decoded into a natural language sentence one word at a time, feeding the output of one time step into the second layer of the LSTM in the subsequent time step. This has been shown in S2VT [35].
Attention mechanisms were initially proposed in [36] and used in the video captioning context by [16]. This allows the selection of relevant temporal segments of a video conditioned on the text-generating recurrent network. Spatial attention over parts of an image is shown by [14]. They use the outputs of the last convolution layer to guide the word generation to look into specific parts of image. They also present a hard-attention mechanism equivalent to reinforcement learning with the reward for selecting the image region proportional to the target sentence. Semantic attention was used to enhance image captioning by independently selecting a list of word attributes [18]. Similarly, [37] and [38] have included video attributes or tags to help generate improved captions. The attribute or tag selection is not trained along with the language model and it becomes challenging to obtain rich attributes or “concepts” for videos that can also categorize actions along with objects.

More recently, video captioning was extended to paragraph generation using independent recurrent networks at the word and sentence level [17]. Hierarchical recurrent networks have also been used to encode the video in an embedding before generating words [15]. They also apply the attention over multiple stages (local, regional and global) which increases the number of learnable parameters.

All described methods are dependent on availability of large scale datasets with video-sentence paired data. [39] demonstrated the use of knowledge transfer from independent language and image data for image captioning. This thesis is loosely inspired by this study in terms of using sentence independent visual concepts to improve the quality of generated
captions. In contrast to traditional soft attention, our model guides the attention using independent temporal “concepts” of the video.

Our work is additionally inspired by the soft attention model for video captioning presented in [16]. We augment it by parameterizing the attention mechanism with a Gaussian distribution over the video length. By learning the normalized mean and sigma values of the distribution, the model removes any dependency on the video duration. This also allows for temporal data augmentation to generate multiple sentences per video. Gaussian attention filters are discussed in [40] but the application is limited to activity classification and their equally spaced attention filters limit the use of attention for word generation.
4.1 Encoder-Decoder models

Initial works that introduced Recurrent Neural Networks (RNNs) for video captioning used a mean pooled [31] feature as the video representation as shown in Figure 11. In this work, the input to the LSTM layer is simply the average of all feature vectors:

\[
\varphi(V) = \frac{1}{n} \sum_{i=1}^{n} v_i
\]  

(21)

Where \( v_i \) is the output of the CNN encoder. By treating all feature vectors with the same importance and averaging them up like this, the mean pool approach limits the model’s capability in recognizing and exploiting the temporal dependencies between frames.

In the following work, Venugopalan et al. [35] introduce an encoder-decoder framework which encodes features from \( f \) frames of a video to the first layer of a two-layer LSTM to
form a latent embedding that is decoded into a natural language sentence by a second LSTM layer (Figure 12). The decoding is done by using the output word from the previous time step along with hidden vectors. One of the main advantages of the encoder-decoder model is the ability to handle variable length sequences.

Pan et al. [15] discovered that by utilizing hierarchy in different stages as shown in Figure 14, their model can effectively learn complex temporal structure of video. This resulted in
higher captioning scores (METEOR, BLEU) in video captioning tasks. Instead of stacking several layers of LSTM (Figure 13), the input sequences \((x_1, x_2, ..., x_T)\) are divided into several chunks \((x_1, x_2, ..., x_n), (x_{1+s}, x_{2+s}, ..., x_{n+s}), ..., (x_{T-n+1}, x_{T-n+2}, ..., x_T)\), where \(s\) is stride which depicts the distance between two adjacent chunks. After feeding into the first LSTM filter, the output will be a sequence of feature vectors \(h_1, h_2, ..., h_{\lfloor T/n \rfloor}\) and input into second LSTM layer. The first layer acts as local temporal structure learner of each subsequence, while the second layer exploits the global temporal structure among subsequences. This method has proved to be extremely successful, yielding robust scores (METEOR, BLEU) in video captioning tasks [15].

*Figure 14 HRNE model. [15]*
4.2 Soft Attention

A simple way to encode a video is by averaging pixels or features across all frames in the video. Most commonly, features are the output of a frame passed into an ImageNet pre-trained CNN. However, this video representation would reduce the model’s ability to learn temporal structure. To avoid this problem, Yao et al. [16] introduced soft attention mechanisms, which can focus on a subset of the input sequence at each time step. Soft attention uses a weighted combination of these frame-level features, where the weights are influenced by the word decoder. Assume we have a sequence of feature vector \( V = \{v_1, v_2, \ldots, v_n\} \), unnormalized relevance score \( e_i^t \) of \( i^{th} \) temporal feature at decoder time step \( t \) is computed by:

\[
e_i^t = w^T \tanh(W_a h_{t-1} + U_a v_i + b_a)
\]  

(22)

Where, \( h_{t-1} \) is the hidden state at the previous time step of the decoder, \( v_i \) is the frame feature vector representation of the \( i^{th} \) frame, and \( w, W_a, U_a, b_a \) are learned parameters. This can be interpreted as an alignment between the encoder and decoder sequence. It
allows the video encoder to selectively weigh on parts of the video. As the frame relevance score is computed using fixed dimension weight matrices, it restricts the exact number of frames in the video. Moreover, given that the average length of videos is only a few seconds in most datasets, it seems counter intuitive to have strong localized attention in such a short duration.

After relevance scores are computed, dynamic weights $\alpha_i^{(t)}$ are obtained by normalizing them:

$$
\alpha_i^{(t)} = \frac{\exp\{e_i^{(t)}\}}{\sum_{j=1}^{n} \exp\{e_j^{(t)}\}}
$$

(23)

Then, the weighted sum of the temporal feature vectors is calculated:

$$
\varphi_t(V) = \sum_{i=1}^{n} \alpha_i^{(t)}v_i
$$

(24)
This research introduces two video captioning architectures with several techniques:

1. Gaussian attention which aims to replace the popular Soft Attention (4.2), with advantages over SA such as fewer parameters and better captioning performance.

2. Attention steering aids the attention mechanism to boost the model’s performance.

3. Shot boundary detection which enables flexible hierarchy of layers instead of a fixed stride as discussed in 4.1.

4. Hierarchical architecture with fixed strides, along with soft hierarchy techniques using shot boundary detection.
5.1 Gaussian Attention

We define the Gaussian Attention (GA) to remove restrictions with the generic soft attention mechanism. The relevance score that weighs the input sequence is modeled with a Gaussian distribution. At each time step, the decoder observes a filtered/weighted encoder sequence. GA weighs the input sequence based on the temporal location and the shape of the distribution is modeled by the mean and standard deviation. We adapt the function to compute a continuous relevance score $e^t$ across the entire input sequence $X = (x_1, x_2, \ldots, x_F)$ at decoder time step $t$ as:

$$
e^t = \sum_{k=1}^{N} \pi_k N(X|\mu_k^t, \Sigma_k^t)$$  \hspace{1cm}(25)$$

Where, each GA is a Gaussian distribution with its unique mean $\mu_k^t$ and covariance matrix $\Sigma_k^t$ at time $t$, $N$ is the number of Gaussians and $\pi_k$ is a mixing coefficient. The mixing coefficients are normalized to sum to one. The input features $X \in \mathbb{R}^{D \times F \times M}$, where $D$ is the number of input modalities, $F$ is the length of each sequence, and $M$ is the dimension of each feature. For example, if the two input modalities of spatial domain and temporal domain are used, we can learn a unique set of Gaussians for each modality by setting $D = 2$. By varying mixing coefficients, mean and covariance of basic Gaussians, the superposition can approximate any continuous function by using sufficient number of Gaussians. Hence with correct parameters, a GA model can achieve the same function as
soft attention. We choose to model independent Gaussians, and replace $\Sigma_k^t$ with a scalar standard deviation, $\sigma^t$ at each time $t$.

Computing the parameters allows the filter to temporally adapt to decoder decisions. With the loss back-propagated at each time step, the mean value of the Gaussian learns to focus on relevant locations of the sequence. Similarly, the standard deviation can learn to extract information from a longer or shorter segment. Thus, the GA formulation makes it adaptive both in terms of location and range. Resource utilization can be optimized as the decoder need not necessarily compute attention over the entire input sequence. The mean and standard deviation are computed as:

$$
\mu^t = \phi(W_\mu h_{t-1} + U_\mu X + b_\mu) \\
\sigma^t = |W_\sigma h_{t-1} + U_\sigma X + b_\sigma|
$$

Where $W_\mu, W_\sigma, U_\mu, U_\sigma, b_\mu, b_\sigma$, are learned weights. We use the activation $\phi(s) = \frac{|s|}{|s| + c}$ for the mean values to scale to range $[0, 1]$ as the input sequence is normalized temporally. The normalization allows the model to compute attention over sequences of varying length. It also reduces the number of learnable weights from $R^{h \times h}$ to $R^{h \times N}$, where $h$ is hidden dimension size of decoder and $N \ll h$. Like soft attention, the attention weights $\alpha^t_k$ at time $t$ for input $X$ are obtained by normalizing the relevance scores. The input to the decoder is a weighted sum of the input $X$ using the attention at time $t$. 
Figure 16 Illustration of the parameterized Gaussian attention model for steering the temporal alignment between the video and word sequence. The caption is generated using a recurrent neural network. For a video, the mean and standard deviation of the distribution is computed based on the outputs of the previous time steps (dotted lines). The curves depict change in the attention over the video based on the word generated in the caption generator.

Modeling the attention filter with a parametric distribution allows the decoder to view inputs with varying duration and hence it is better at exploiting the temporal structure of an input sequence. The parametric attention has the capability to sense the complete encoder sequence if required. This is important in a translation-like task where the generated word may hold relevance throughout the video. For example, after the word “man” in Figure 16, the model learns to expand the attention to allow the caption generator to view the entire input as the associated visual feature of “man” appears in the entire video.
5.2 **Attention steering**

Traditional attention models are associated with a set of weight matrices that are learned during training. During test time, the weight matrices guide the attention and hence limit the attention mechanisms by prior training statistics. This is important as most video captioning datasets cannot possibly have a comprehensive representation of all activities and objects. If a certain semantic action/object like “cutting an apple” is more likely to appear in beginning of the video in the training set, the learned model may expect similar trends in test videos.

To obtain video specific attention, we both allow the network to “watch” the full video before attending to specific portions, and we introduce temporal concepts along the portions of the video. One way to encode a summary of an entire video can be done using an LSTM as shown in [35]. LSTMs have the capability to retain segments of the video but would have to remember both the context of the overall video and the relevant frames. With this restriction, it becomes difficult for the latent representation of the video (we use a dimension of 512) to have detailed attention, and surely would struggle for long videos. To overcome this restriction, we introduce temporal attention steering that guides the attention based on the feature statistics of the test video.

We further investigate the use of word label embeddings of objects present in video frames as temporal visual features. We use an ImageNet classifier trained on 4k classes [41] represented using a Glove [42] word embedding. Representing a large number of objects is important for “in-the-wild” videos. A bottom-up grouping strategy [41] is applied to the categories to deal with the problems of over-specific classes.
In reality, a sentence is described by both the objects and the whole scene as the context. Distinguishing individual objects from others in a scene, especially when there are multiple objects of different categories can be highly challenging. Hence, EdgeBox [43] is used to obtain proposal bounding box regions within each frame of a video. For the top 95% of all bounding boxes, we compute Glove word embedding of the ImageNet 4K CNN classes. These embedded word vectors are mean pooled to obtain a frame-level representation. We discover that the mean pooled class label embedding is rich in semantic information and is closer to the words in the ground truth sentence. Moreover, use of word embedding reduces the feature dimension from 4K to 300. This design choice reduces the number of learnable parameters substantially. As a complementary or alternative approach to temporal word concepts, one could use frame CNN features directly.

Figure 17. Frame level features are weighted based on the relevance map and assists in guiding attention to video regions. $W_t$ and $W_{t-1}$ are words at times $t$ and $t-1$, $h_{t-1}$ is RNN hidden state.
5.3 Video2vec representation

In addition to the steering mechanism, an embedded vector representation of the entire video is input into the captioning model (right input in Figure 19). Just as word concepts capture the temporal object information, video level features (about the actions and activities in the video) are equally important. An embedding function \( f: V \rightarrow S_v \), that maps a video \( V \) with frames \( (v_1, v_2, \ldots, v_n) \) into a representation \( S_v \) is learned. We refer this transformation as Video2Vec.

**Video2Vec-Activity**: To learn powerful action and motion concepts, we use a recent activity classification dataset - ActivityNet [44], on human activity understanding that covers a wide range of complex daily activities. It is comprised of 849 video hours in over 200 activity classes. As these videos were collected from online video sharing sites they are excellent to transfer learned features for MSVD and MSR-VTT datasets which are also based on Youtube videos. The labeled videos are used to train a standard video-based activity classifier. We train two independent models using RGB (3-color channels) and optical flow inputs. Features before the loss layer are used as Video2Vec-Activity representation and we fine-tune the last fully-connected layer during caption generation.

**Hierarchy over Gaussian Attention**: As discussed in 4.1, the recently proposed hierarchical neural encoder [15] technique efficiently captures temporal dependencies in videos. Hence, we integrate it with our GA model and term it as Hierarchy over Gaussian Attention (HGA). The hierarchy of recurrent layers adds more non-linearity to the GA model. The hidden state of LSTM in layer \( l-1 \) at the last time step is the input to layer \( l \).
This ensures easy back-propagation of loss compared with a simple layer stacking by reducing the number of steps the loss back-propagates. The first layer learns local temporal dynamics within short clips and the second layer learns the difference between these short clip sequences. The output at the last time step of the second layer is a vector representation for the entire video. We further adapt [15] by replacing soft attention with Gaussian attention between all recurrent layers.
5.4 Shot boundary detection

Features extracted from CNN models have proven to be useful in cut-transition boundary detection between two shots in a video stream [45]. Given $\alpha_i$ and $\alpha_j$ are two CNN feature vectors of two consecutive frames, the cosine distance $\Delta (i, j)$ between them can be calculated as follows:

$$\Delta(i, j) = \cos(\alpha_i, \alpha_j) = \frac{\alpha_i \cdot \alpha_j}{|\alpha_i||\alpha_j|}$$

(28)

$\Delta (i, j) \in [0,1]$, with higher values having higher probability of a boundary cut. For example, one could experimentally determine a threshold $\zeta$ where a boundary exists when $\Delta (i, j) > \zeta$. Unlike Euclidean distance, the cosine distance needs no additional normalization steps. Xu et al. [45] determined this distance is effective in the cut-transition detection task. Our results concur and we employ it to detect the boundary to facilitate the soft hierarchy layer.
5.5 **Multi-stream Hierarchical Boundary Model**

Referring to Figure 18, the encoding stage takes in a given video stream, whereby the first layer takes in local features \((x_1, x_2, \ldots, x_n)\), and outputs two sequence vectors: 1) equal spaced \((w_1, w_2, \ldots, w_p)\); and 2) clip level \((z_1, z_2, \ldots, z_q)\). The equal spaced output layer gets \(p\) outputs from first layer with \(p = n/k\) (\(n\) is number of input features and \(k\) is designed stride value). The clip level output layer utilizes information on shot boundaries guided by a learned vector based on the cosine distance:

\[
z_i = y_i \cdot (\Delta(i,j) \cdot W_{yd} + b_{yd})
\]  

where \(W_{yd}\) and \(b_{yd}\) are learned weights and bias, \(y_i\) is output at each time step of first layer. As illustrated in Figure 18 the video is encoded through a combination of equally spaced and clip level feature representations. The fusion of local (frame) level, hierarchy (equally spaced) and clip (detected boundaries) level is input to the caption decoder. At each time step, the model adapts the boundary weights to extract information from the relevant segments of the video thus being extremely efficient in encoding complex video sequences.
Figure 18 Overview of the Multi-stream Hierarchical Boundary Model for video captioning.
5.6 Gaussian steering model

This video captioning framework has three main components - Attention Steering, Video2Vec encoder and Gaussian attention based sentence generator as shown in Figure 19 (left, right and center). The sentence generation engine takes in input from all three to generate word sequences. Recurrent Neural Networks (RNN) are a natural choice for generating sequences such as natural language sentences. RNNs suffer from vanishing and exploding gradient problems when learning long sequences. Hence, we use the LSTM variant of RNNs to learn sentence generation as it is known to learn sequences with both short and long temporal dependencies.

Figure 19 Overview of Gaussian steering model for video captioning.
5.7 Loss function

At a high level, our caption generation decoder (RNN) layer can be defined as:

\[
\begin{bmatrix}
    p(y_t | y_{<t}, V) \\
    h_t \\
    c_t
\end{bmatrix} = \psi(h_{t-1}, c_{t-1}, y_{t-1}, V)
\]  

(30)

with:

- \(y_t\) is the output of decoder at time \(t\).
- \(h_t, h_{t-1}\) are the hidden state of RNN layer at time step \(t\) and \(t-1\), respectively.
- \(c_t, c_{t-1}\) are cell states of RNN layer at time \(t\) and \(t-1\).
- \(V\) is encoder representation.

Once the hidden state \(h_{t-1}\) is computed, the probability distribution of the next word is obtained by:

\[p_t = \text{softmax}(U_p \tanh(W_p [h_t, \varphi_t(V), E[y_{t-1}] + b_p]) + d)\]  

(31)

where \(U_p, W_p, b_p, d\) are learnable parameters, \(h_t\) is the output of RNN layer at time step \(t\), \(\varphi_t(V)\) is the decoder representation and \(E[y_{t-1}]\) is the embedding of the word \(y_{t-1}\).

Then we have the objective (loss) function:

\[\text{loss} = \frac{1}{T} \sum_{t=1}^{T} p_t\]  

(32)
As can be observed, this loss focuses on optimizing $l$-gram accuracy (BLEU-1) of generated captions, thus limiting its capability. Further research can improve this by apply other loss computation techniques like policy gradients from metric score like METEOR or CIDEr as suggested in [46].
Chapter 6  Dataset and training details

6.1 Dataset

We choose the Microsoft Video Description Dataset (MSVD) [19], the newly released Microsoft Research - Video to Text (MSR-VTT) [47] and Montreal Video Annotation Dataset (M-VAD) [20], to evaluate the proposed model. MSVD is a common dataset and is used to benchmark most recent proposals recently ([15, 17, 35]). Standard train, validation and test splits were used for all datasets and detailed statistics are listed in Table 1.

We only use the video and captions for all datasets except MSR-VTT which also includes metadata in the form of 20 coarse level categories for each video.

Table 1. Video-sentence pair dataset statistics.

<table>
<thead>
<tr>
<th></th>
<th>MSVD</th>
<th>MSR-VTT</th>
<th>M-VAD</th>
</tr>
</thead>
<tbody>
<tr>
<td># sentences</td>
<td>80,827</td>
<td>200,000</td>
<td>54,997</td>
</tr>
<tr>
<td>#sentences per video</td>
<td>~42</td>
<td>20</td>
<td>~1-2</td>
</tr>
<tr>
<td>Vocab size</td>
<td>9,729</td>
<td>24,283</td>
<td>16,344</td>
</tr>
<tr>
<td>Average length</td>
<td>10.2s</td>
<td>14.8s</td>
<td>5.8s</td>
</tr>
<tr>
<td># Train video</td>
<td>1,200</td>
<td>6,513</td>
<td>36,921</td>
</tr>
<tr>
<td>#Validation video</td>
<td>100</td>
<td>497</td>
<td>4,651</td>
</tr>
<tr>
<td>#Test video</td>
<td>670</td>
<td>2,990</td>
<td>4,951</td>
</tr>
</tbody>
</table>
6.2 Training details

Each video frame is passed through the 152-layer ResNet CNN model [4] pre-trained on ImageNet data, where the $1\times2048$ vector from the last pooling layer-pool5 is used as frame feature representation. In our HGA model, the inputs to the first LSTM layer are 12 frame clips and the output at the last time step is the input to the second layer.

We use the PTBTokenizer in the Stanford CoreNLP tools [48] to pre-process all words in captions. This involved converting to lower case and removing punctuation to tokenize the sentences. We use captions only from the training and validation set to generate the vocabulary. A one-hot vector encoding of the vocabulary is used to represent each word as a vector. For Video2Vec object concepts and MSR-VTT video categories, we use 300-dimension Glove embedding [42] to obtain word vector representations.

The architecture is implemented in TensorFlow [49]. During training, ADAM optimization is used to minimize the negative log likelihood loss. The learning rate is $2\times10^{-4}$ and we use decay parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$) as reported in [50]. The hidden dimension of LSTM layers in HGA is 1024 and for the sentence generation layer is 384 empirically. We employ a Dropout [51] of 0.5 on the output of all LSTM layers. The mini-batch size is 100 and all models are trained for 20 epochs. The training time took three hours for the simple baseline model (GA-5) and up to ten hours for a complex model (HGA) with MSVD dataset. On the contrary, the testing time took five seconds for a batch size of 100 with the baseline model and up to 50 seconds with the HGA model.
Hyperparameters are selected by running tests on the validation set.

6.3 Beam Search

The LSTM generates a word at each time step. Instead of a greedy search for the word with best probability, beam search is employed at test time to have more variability in words and sentences. During the first time step, the $k$ top words are used to generate the next words at time step $t+1$. For all other time steps, the top $k$ sentences are used to generate the next word at time step $t+1$. This ultimately yields $k$ candidate sentences each scored with an overall probability as the sum of probability of all words in that sentence at each time step.

For example, in Figure 20, at time step $T=0$, three words are selected by distributed possibilities over dictionaries: “a”, “the”, “red”. At time step $T=1$, each embedding of these words will combine with the current model’s parameters and produce new possible words at $T=1$. Then, the three words with the highest combination probabilities are selected, and the process continues. Finally, we have a map of words as depicted in Figure 21, where if one follows the arrow from left to right, we have the three final sentences with the highest possibilities.
Empirically, a beam-width of 10 with MSVD and 20 with MSR-VTT performs the best. We conjecture that the large vocabulary size for MSR-VTT required a higher beam width. This is in agreement with [37].
Chapter 7  Evaluation metrics

Quantitative evaluation was performed using the Microsoft COCO caption evaluation tool to make our results directly comparable with other studies. For evaluation, we use standard metrics- BLEU [53], METEOR [54], CIDEr [55] and ROUGE [56] to score a predicted sentence against all ground truth sentences. Generally, the generated sentence correlates well with a human judgment when the metrics are high as they measure the overall sentence meaning and fluency. Table 2 [57] shows the underlying idea of each metric score. In most recent publications [15-17, 35], METEOR, BLEU, CIDEr and ROUGUE-L score are metrics of choice since they better align with human judgement than other methods. Hence, in our experiments we employed them to compare with other architecture’s results. We report all scores in percentages. Section from 7.1 to 7.4 will discuss how these metric scores are calculated.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Proposed to evaluate</th>
<th>Underlying idea</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLEU</td>
<td>Machine translation</td>
<td>n-gram precision</td>
</tr>
<tr>
<td>ROUGE</td>
<td>Document summarization</td>
<td>n-gram recall</td>
</tr>
<tr>
<td>METEOR</td>
<td>Machine translation</td>
<td>n-gram with synonym matching</td>
</tr>
<tr>
<td>CIDEr</td>
<td>Image description generation</td>
<td>\textit{tf-idf} weighted n-gram similarity</td>
</tr>
<tr>
<td>SPICE</td>
<td>Image description generation</td>
<td>Scene-graph synonym matching</td>
</tr>
</tbody>
</table>
7.1 BLEU

BLEU (BiLingual Evaluation Understudy) [53] is a widely use metric in machine translation task. BLEU computes the overlap of n-gram between the predicted and ground truth sentences. Corpus-level clipped n-gram precision between sentences is calculated as follows:

\[
CP_n(C, S) = \frac{\Sigma_i \Sigma_k \min (h_k(c_i), \max_{j \in m} h_k(s_{ij}))}{\Sigma_i \Sigma_j h_k(c_i)}
\] (33)

Where \( S \) is the set of reference captions, \( C \) is predicted sentence, \( k \) indicates the set of n-grams of length \( n \). Observed that \( CP_n \) favors short sentences. Thus the brevity penalty is provided as:

\[
b(C, S) = \begin{cases} 
1 & \text{if } l_c > l_s \\
\exp \left( 1 - \frac{l_s}{l_c} \right) & \text{if } l_c \leq l_s 
\end{cases}
\] (34)

With \( l_c \) is total length of predicted sentences, \( l_s \) is the closest reference length among ground truth sentences.

Then the overall BLEU score is computed using geometric mean of the individual n-gram precision:

\[
BLEU_N(C, S) = b(C, S) \exp \left( \sum_{n=1}^{N} w_n \log CP_n(C, S) \right)
\] (35)

Example

Ground truth sentence: Israeli officials are responsible for airport security

Assume we have two models which produced the following two sentences:

Predicted sentence 1: **Israeli officials** responsibility of **airport** safety

Predicted sentence 2: **airport security** **Israeli officials are responsible**.
Table 3 demonstrate the steps to calculate the BLEU score

<table>
<thead>
<tr>
<th>Metric</th>
<th>Predicted sentence 1</th>
<th>Predicted sentence 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision(1-gram)</td>
<td>3/6</td>
<td>6/6</td>
</tr>
<tr>
<td>Precision(2-gram)</td>
<td>1/5</td>
<td>4/5</td>
</tr>
<tr>
<td>Precision(3-gram)</td>
<td>0/4</td>
<td>2/4</td>
</tr>
<tr>
<td>Precision(4-gram)</td>
<td>0/3</td>
<td>1/3</td>
</tr>
<tr>
<td>Brevity penalty</td>
<td>6/7</td>
<td>6/7</td>
</tr>
<tr>
<td><strong>BLEU</strong></td>
<td><strong>0%</strong></td>
<td><strong>52%</strong></td>
</tr>
</tbody>
</table>
7.2 METEOR

METEOR (Metric for Evaluation of Translation with Explicit ORdering) [54] is a metric score calculated by creating an alignment between the words of a predicted sentence and ground truth sentence, aiming at 1:1 correspondence.

Given a set of alignments, $m$, the METEOR score is the harmonic mean of precision $P_m$ and recall $R_m$ between best scoring reference and candidate. After having $P_m, R_m$, we calculate the harmonic mean as below:

$$P_m = \frac{|m|}{\sum_k h_k(c_i)}$$  \hspace{1cm} (36)

$$R_m = \frac{|m|}{\sum_k h_k(s_{ij})}$$  \hspace{1cm} (37)

$$F_{mean} = \frac{P_m R_m}{\alpha P_m + (1 - \alpha) R_m}$$  \hspace{1cm} (38)

To take account of long matches between candidate sentence and reference ones, METEOR compute the Penalty, with $ch$ is number of chunks and $m$ is number of alignments (number of unigram matched):

$$Pen = \gamma \left( \frac{ch}{m} \right)$$  \hspace{1cm} (39)

Finally, the METEOR score for the given alignment is calculated by:

$$METEOR = (1 - Pen) F_{mean}$$  \hspace{1cm} (40)
For example, we calculate the METEOR score between predicted sentence and the ground truth as below:

Ground truth: the **Iraqi weapons** are to be handed over to the **army** within **two weeks**.

Predicted sentence: in **two weeks Iraq’s weapons** will give **army**.

We have $P_m = \frac{5}{8} = 0.625$, $R_m = \frac{5}{14} = 0.357$, and $F_{\text{mean}} = 0.3731$.

$Pen = 0.5(0.5)^3 = 0.0625$, then we have $METEOR = F_{\text{mean}} (1 - Pen) = 0.3498$


7.3 CIDEr

CIDEr (Consensus-based Image Description Evaluation) [55] is a metric designed specifically for evaluation in image captioning tasks. It assesses the consensus in image captions by using Term Frequency Inverse Document Frequency (TF-IDF) weighting for each n-gram.

CIDEr calculates TF-IDF weighting $g_k(s_{ij})$ for each n-gram $w_k$ by:

$$g_k(s_{ij}) = \frac{h_k(s_{ij})}{\sum_{w_l \in \Omega} h_l(s_{ij})} \log \left( \frac{|I|}{\sum_{l_p \in I} \min\left(1, \Sigma_{q \in I} h_k(s_{pq})\right)} \right)$$

(41)

Where $\Omega$ is the set of all n-grams and $I$ is collection of all images in the dataset, $h_k(s_{ij}), h_k(c_i)$ is the number of times an n-gram $w_k$ appears in ground truth sentence $s_{ij}$ or predicted sentence $c_i$.

The CIDEr$_n$ score for n-grams of length $n$ is calculated by using the average cosine similarity between predicted sentence and ground truth sentences, which consider both precision and recall:

$$CIDEr_n(c_i, S_t) = \frac{1}{m} \sum_j \frac{g^n(c_i) \cdot g^n(s_{ij})}{\|g^n(c_i)\| \|g^n(s_{ij})\|}$$

(42)

Finally, we have scores for n-gram of varying lengths:
\[ CIDEr(c_i, S_i) = \sum_{n=1}^{N} w_n CIDEr_n(c_i, S_i) \]
7.4 ROUGE-L

ROUGE [56] is a group of metrics proposed to evaluate text summarization algorithms. In this research we use ROUGE-L, a metric score based on the Longest Common Subsequence (LCS). LCS is a collection of words occur in two sentences with the same order. Assume we have length \( l(c_i, s_{ij}) \) of the LCS between two sentences, ROUGE-L is produced by calculating F-measure:

\[
R_t = \max_j \frac{l(c_i, s_{ij})}{|s_{ij}|} \tag{44}
\]

\[
P_t = \max_j \frac{l(c_i, s_{ij})}{|c_i|} \tag{45}
\]

\[
ROUGE_L(c_i, S_t) = \frac{(1 + \beta^2)R_tP_t}{R_t + \beta^2P_t} \tag{46}
\]
Chapter 8  
Results and discussion

8.1 MHB results

Table 4. MSVD caption evaluation results with MHB model.

<table>
<thead>
<tr>
<th>Method</th>
<th>METEOR</th>
<th>BLEU-4</th>
<th>CIDEr</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2VT[35]</td>
<td>29.8</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SA[16]</td>
<td>29.6</td>
<td>41.9</td>
<td>51.67</td>
<td>-</td>
</tr>
<tr>
<td>p-RNN[17]</td>
<td>32.6</td>
<td>49.9</td>
<td>65.8</td>
<td>-</td>
</tr>
<tr>
<td>HRNE Att[15]</td>
<td>33.1</td>
<td>43.8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MHB w/o GA</td>
<td>30.2</td>
<td>39.8</td>
<td>62.0</td>
<td>65.4</td>
</tr>
<tr>
<td>MHB w/o Bdr</td>
<td>32.5</td>
<td>42.3</td>
<td>68.6</td>
<td>68.2</td>
</tr>
<tr>
<td>MHB w/LSTM</td>
<td>32.9</td>
<td>42.3</td>
<td>70.4</td>
<td>68.6</td>
</tr>
<tr>
<td>MHB</td>
<td><strong>33.2</strong></td>
<td>43.0</td>
<td><strong>71.1</strong></td>
<td>68.7</td>
</tr>
</tbody>
</table>

Table 4 compares MHB with recently published results on the MSVD dataset. S2VT [35] uses a two layer LSTM in an encoder-decoder framework. SA uses an attention mechanism over temporal features and feed into a LSTM decoder. Both p-RNN [17] and HRNE [15] utilize hierarchy in different stages, decoder or encoder stage respectively. Table 4 shows that our results on MSVD are competitive among other methods although we do not use other strong features like C3D (p-RNN) and use fewer frames (80 vs. 160 in HRNE).
We conduct various ablation studies to understand the effect of different components of the MHB framework. We find: 1) Using LSTM cells in the decoder stage achieve similar results as RHN cells but they take longer to converge; 2) RHN depth $l=3$ performs better than depth $l=5$, indicating that the hierarchical setting may be complex enough for the video captioning task; 3) We get a lower METEOR score (32.5) for MSVD captioning without the boundary vectors $z_l$. This proves the effectiveness of boundary encoding for representing a video; and 4) Without using a SA in place of GA in the caption decoder stage, the METEOR scores drop from 33.2 to 30.2 for MSVD captioning thus validating its use.

Table 5. MSR-VTT results on held-out test set.

<table>
<thead>
<tr>
<th>Method</th>
<th>METEOR</th>
<th>BLEU-4</th>
<th>CIDEr</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dong et al.[37]</td>
<td>26.9</td>
<td>39.3</td>
<td>45.9</td>
<td>58.3</td>
</tr>
<tr>
<td>Multimodal[58]</td>
<td>27.0</td>
<td>38.1</td>
<td>41.8</td>
<td>59.7</td>
</tr>
<tr>
<td>Shetty[32]</td>
<td>27.7</td>
<td>41.1</td>
<td>46.4</td>
<td>59.6</td>
</tr>
<tr>
<td>MHB(ours)</td>
<td>27.3</td>
<td>37.8</td>
<td>42.6</td>
<td>58.8</td>
</tr>
</tbody>
</table>

Table 5 and Table 6 compare the MHB model with recent results on MSR-VTT and MVAD datasets. [37] used a tagging embedding to enrich the LSTM input and re-rank generated sentences by their relevance to a video. Multimodal [58] utilized a model based on S2VT [35] with multi modal input features. MHB results are competitive on MVAD dataset, but are disadvantaged due to poor alignment of the ground truth caption with the video frames which sometimes result in confusing cut-transition boundaries.
Table 6 Comparison of captioning results on MVAD dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>METEOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2VT[35]</td>
<td>5.6</td>
</tr>
<tr>
<td>HRNE[15]</td>
<td>5.8</td>
</tr>
<tr>
<td>HRNE with attention [15]</td>
<td>6.8</td>
</tr>
<tr>
<td>Venugopalan et al. [59]</td>
<td>6.8</td>
</tr>
<tr>
<td>HBANE+Resnet50+C3D[60]</td>
<td>7.3</td>
</tr>
<tr>
<td>MHB</td>
<td>6.6</td>
</tr>
</tbody>
</table>

Figure 22 demonstrates a learned boundary vector for an example video. Aided by on the learned cosine similarity, our model can recognize cut-transitions between multiple shots in video. This generates an efficient video representation.

Figure 22 Example boundary attention vector where the “dips” indicate video boundaries in [0 – 1] normalized video.
8.2 HGA results

8.2.1 Performance on MSVD

Table 7 reports current captioning results (top half) vs. variations on our model (bottom half) on the MSVD dataset. Our baseline model (Baseline GA-5) is a Gaussian attention with five Gaussians. The addition of a hierarchical model (+HGA) shows significant improvement. The HGA model learns powerful motion features that a simple attention is unable to capture. As recommended in [37], we test a baseline variant with BLEU-4 score included in the caption loss (BLEU regularization). The BLEU score is computed on the validation set and regularized with the loss after each mini-batch. Though it significantly improves BLEU scores, other scores are not much affected and we notice that sentence fluency degrades as well.

The addition of Video2Vec-Activity (+RGB, OF) further helps with METEOR scores due to importance of motion features. The highest METEOR score that we achieve is 33.1% which matches the state-of-the-art. We achieve a high BLEU-1 score with the baseline model and the hierarchy seems to hurt the n-gram metric.
Table 7 MSVD caption evaluation results on the held-out test set.

<table>
<thead>
<tr>
<th>Method</th>
<th>METEOR</th>
<th>BLEU-1</th>
<th>BLEU-2</th>
<th>BLEU-3</th>
<th>BLEU-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP[31]</td>
<td>29.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>33.3</td>
</tr>
<tr>
<td>S2VT[35]</td>
<td>29.8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SA[16]</td>
<td>29.6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>41.9</td>
</tr>
<tr>
<td>p-RNN[17]</td>
<td>32.6</td>
<td>81.5</td>
<td><strong>70.4</strong></td>
<td><strong>60.4</strong></td>
<td><strong>49.9</strong></td>
</tr>
<tr>
<td>HRNE Att [15]</td>
<td>33.1</td>
<td>79.2</td>
<td>66.3</td>
<td>55.1</td>
<td>43.8</td>
</tr>
<tr>
<td>Baseline GA-5</td>
<td>31.5</td>
<td>80.4</td>
<td>66.6</td>
<td>54.5</td>
<td>42.8</td>
</tr>
<tr>
<td>+HGA</td>
<td>32.8</td>
<td>79.1</td>
<td>65.8</td>
<td>54.8</td>
<td>43.9</td>
</tr>
<tr>
<td>BLEU reg</td>
<td>30.8</td>
<td><strong>81.6</strong></td>
<td>68.3</td>
<td>55.2</td>
<td>42.4</td>
</tr>
<tr>
<td>+RGB, OF</td>
<td><strong>33.1</strong></td>
<td>77.5</td>
<td>64.1</td>
<td>53.4</td>
<td>43.0</td>
</tr>
</tbody>
</table>
8.2.2 Performance on MSRVTT

Caption evaluation scores for our models on the MSR-VTT dataset are reported in Table 8 and Table 9. Each model is trained end-to-end and we compare our approach with recent results. A single layer GA performs better than the mean pooled video frame input features. The HGA model adds hierarchy features to a single layer GA model and hence is better at learning temporal dependencies. The significance of Gaussian attention is shown by comparison of HGA with and without attention. This has better performance than a weighted average through an attention mechanism. To study the importance of temporal steering (STE) and Video2Vec-Activity (RGB and OF) features, we also input these as features to the captioning model. These inputs have positive impacts on the evaluation metrics. The addition of activity features show clear improvement over the baseline HGA. The OF features yield slightly improved scores over RGB. This indicates that motion/activity features from the ActivityNet dataset generalize well to other datasets.

Table 8 HGA results on the held out MSR-VTT test set.

<table>
<thead>
<tr>
<th>Method</th>
<th>METEOR</th>
<th>BLEU-4</th>
<th>CIDEr</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only GA 1-layer</td>
<td>25.6</td>
<td>34.6</td>
<td>37.4</td>
<td>57.4</td>
</tr>
<tr>
<td>HGA (w/o att)</td>
<td>26.6</td>
<td>36.0</td>
<td>38.9</td>
<td>58.4</td>
</tr>
<tr>
<td>HGA</td>
<td><strong>27.4</strong></td>
<td><strong>38.8</strong></td>
<td><strong>43.4</strong></td>
<td><strong>59.1</strong></td>
</tr>
</tbody>
</table>
We observe that across all features the scores did not change significantly when trained without word features loss from [37]. However, it helped the model to converge faster. While generating the vocabulary from the training captions, we note that out of total 24,282 words, 10,155 words appear just once and 3,211 words twice. From the vocabulary, 4,716 words were not part of the Glove 400K dictionary. Such issues add to challenges of the language model. Similar trends appear in other datasets as well.

Fusion based models: Although the METEOR score does not improve with a combination of RGB and optical flow features, all other metrics show improvement. It also indicates that either of the features are sufficient to capture the activity information. We also use the Glove embedded video category label (CAT) available for all videos. The combined model is trained by concatenating the features before input to the LSTM. We note that the categories are the ground truth labels that are part of the original dataset and hence are better than any features generalized from another dataset.

Experiments are run on the HGA model to compare soft and Gaussian attentions. The HGA-only model can be interpreted as a three-layer LSTM with the first two hierarchical layers as the video encoder and the last layer as the sentence generator or word decoder.
Table 9 MSR-VTT results on the held-out test set. We compare with recent entries in the MSR Video to Language Challenge.

<table>
<thead>
<tr>
<th>Method</th>
<th>MET</th>
<th>B-1</th>
<th>B-2</th>
<th>B-3</th>
<th>B-4</th>
<th>CIDEr</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dong et al. [37]</td>
<td>26.9</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>39.3</td>
<td>45.9</td>
<td>58.3</td>
</tr>
<tr>
<td>Multimodal [58]</td>
<td>27.0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>38.3</td>
<td>41.8</td>
<td>59.7</td>
</tr>
<tr>
<td>Shetty et al. [61]</td>
<td>27.7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>41.1</td>
<td>46.4</td>
<td>59.6</td>
</tr>
<tr>
<td>Mean pool [31]</td>
<td>25.4</td>
<td>75.3</td>
<td>60.4</td>
<td>46.4</td>
<td>34.1</td>
<td>35.8</td>
<td>57.7</td>
</tr>
<tr>
<td>Ours</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HGA</td>
<td>27.4</td>
<td>79.7</td>
<td>64.8</td>
<td>51.1</td>
<td>38.8</td>
<td>43.4</td>
<td>59.1</td>
</tr>
<tr>
<td>STE</td>
<td>27.6</td>
<td>79.8</td>
<td>64.4</td>
<td>50.1</td>
<td>37.9</td>
<td>43.4</td>
<td>59.2</td>
</tr>
<tr>
<td>RGB</td>
<td>27.6</td>
<td>79.7</td>
<td>64.6</td>
<td>50.8</td>
<td>38.6</td>
<td>42.8</td>
<td>58.9</td>
</tr>
<tr>
<td>OF</td>
<td>27.7</td>
<td>79.5</td>
<td>64.8</td>
<td>50.9</td>
<td>39.0</td>
<td>43.8</td>
<td>59.6</td>
</tr>
<tr>
<td>RGB, OF</td>
<td>27.7</td>
<td>79.9</td>
<td>64.9</td>
<td>51.0</td>
<td>39.2</td>
<td>43.5</td>
<td>59.2</td>
</tr>
<tr>
<td>RGB, OF, CAT</td>
<td><strong>28.2</strong></td>
<td><strong>80.3</strong></td>
<td><strong>66.1</strong></td>
<td><strong>52.5</strong></td>
<td><strong>40.5</strong></td>
<td><strong>45.3</strong></td>
<td><strong>60.4</strong></td>
</tr>
</tbody>
</table>

We replace soft attention with GA at multiple layer combinations. Results are reported in Table 10. Adding GA at more layers seem to help focus on relevant inputs and features, thus produce better captions. This is illustrated by better METEOR and BLEU scores after replacing soft attention layer by layer. Attention on the middle HGA layers can be viewed as the weighted sum of the encoded outputs of video clips input to the first layer. Attention is most important at the word decoder (layer 3) as it not only finds relevant segments in the video but also relevant HGA encoded features based on generated words. These results
also show that our HGA model is capable of surpassing the HRNE model in captioning task (the first model with no replacing of GA is our HRNE implementation).

Table 10 Comparing Gaussian attention at different layers for MSR-VTT test set. Adding GA show clear improvement over SA and attention is most important at the word generation layer.

<table>
<thead>
<tr>
<th>Layer with replacing SA with GA</th>
<th>MET</th>
<th>B-1</th>
<th>B-2</th>
<th>B-3</th>
<th>B-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>26.7</td>
<td>77.7</td>
<td>62.6</td>
<td>48.4</td>
<td>36.1</td>
</tr>
<tr>
<td>3</td>
<td>27.3</td>
<td>78.9</td>
<td>63.6</td>
<td>49.8</td>
<td>38.1</td>
</tr>
<tr>
<td>3,2</td>
<td>27.4</td>
<td>79.3</td>
<td>64.5</td>
<td>50.8</td>
<td>38.8</td>
</tr>
<tr>
<td>3,2,1(HGA)</td>
<td><strong>27.4</strong></td>
<td><strong>79.7</strong></td>
<td><strong>64.8</strong></td>
<td><strong>51.1</strong></td>
<td><strong>38.8</strong></td>
</tr>
</tbody>
</table>
8.3 Loss curves

Figure 23, Figure 24 and Figure 25 illustrate the loss curves for the training of the baseline model (GA-5), HGA model and MHB model, respectively. As can be observed, with hierarchical architecture employed in HGA and MHB, the models converge faster than the baseline model (GA-5).

Figure 23. Loss value of GA-5 model over iterations.

Figure 24. Loss value of HGA model over iterations.
Figure 25. Loss value of MHB model over iterations.
8.4 **Analysis of Gaussian Attention**

Table 11: Performance evaluation with number of Gaussian filters 532 for attention on the MSVD test set.

<table>
<thead>
<tr>
<th>#Gaussians</th>
<th>METEOR</th>
<th>BLEU-1</th>
<th>BLEU-2</th>
<th>BLEU-3</th>
<th>BLEU-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30.7</td>
<td>76.3</td>
<td>62.3</td>
<td>50.3</td>
<td>39.0</td>
</tr>
<tr>
<td>3</td>
<td>31.2</td>
<td>77.6</td>
<td>64.1</td>
<td>53.0</td>
<td>42.1</td>
</tr>
<tr>
<td>5</td>
<td>31.5</td>
<td><strong>80.4</strong></td>
<td>66.6</td>
<td>54.5</td>
<td><strong>42.8</strong></td>
</tr>
</tbody>
</table>

Since GA allows the model to focus on segments of the input, we train baseline GA models with 1, 3 and 5 Gaussians. Results are reported in Table 11. More attention curves allow the model to view specific but multiple regions of the input by increasing the number of learnable parameters. We observed exploding gradient problems with higher number of Gaussians as the standard deviation starts to approach zero.
Figure 26. Gaussian attention visualization for sample videos from MSVD. Distribution focuses on relevant video segment based on key words (bold) in the sentence. For the word adding, relevant activity is in the starting of video, hence the mean of the distribution is close to 0. X-axis ranges from 0 - 1 normalized temporal video location and Y-axis is normalized attention weight $\alpha_t$.

Figure 26 shows words from generated sentences along with the temporal Gaussian attention distributions generated on sample test MSVD videos. The distribution shows the adaptable nature of Gaussian attention. Even though the videos are short, at certain times the model needs to attend to different parts of the video. We anticipate that with longer and more complex videos, a higher number of Gaussians would be required. More broadly, these results indicate that the Gaussian attention is not restricted by the video duration since the videos are normalized temporally.
Since most of the videos in the MSVD dataset are very short (average duration of 10.2 seconds), it is difficult to do temporal analysis of Gaussian attention. We conduct experiments by combining multiple distinct videos temporally and using the resulting (longer and more complex) video as a test sample. This also gives an example where the Gaussian attention can handle videos of varying duration. For example, we combine 3 test clips (first two with 16 frames and last with 18 frames to form a total 50 frame clip) to form one long video clip.

Table 12 shows the Gaussian attention outputs, with different colors illustrating different Gaussian shape. The model generates attention curves while generating a word at each time step. The model is comprised of five Gaussians and hence five different curves at each time step. We notice that the Gaussians change the shape to attend to the relevant video clip. For example, the word “pool” in example 1, has one of the Gaussians focused in the starting on the video which has the relevant features. Similarly, with the word “man”, the means of Gaussian shapes tend to move toward the end of the video. In general, we notice that majority of attention is over all three videos. Since the model is not trained on such complex combinations, the overall sentence does not seem good, but it can be used to explore relationships between Gaussian shape and the generated caption.

Figure 27 Complex clip combined from 3 totally different clips.

Predicted sentences:

Only clip 1 - A dog is in the pool (GT - a dog drinks from a swimming pool)
Only clip 2 - A man is mixing a bowl (GT-a person is forming a wad out of a mixture in a bowl)

Only clip 3 - A man is talking on a phone (GT - a man is talking on a phone)

Three clips combined - A man is playing with a pool

Table 12 Illustration of Gaussian shape correspond with word in the caption sentence.

<table>
<thead>
<tr>
<th>A</th>
<th><img src="image" alt="Table 12 Illustration of Gaussian shape correspond with word in the caption sentence." /></th>
</tr>
</thead>
<tbody>
<tr>
<td>man</td>
<td><img src="image" alt="Table 12 Illustration of Gaussian shape correspond with word in the caption sentence." /></td>
</tr>
<tr>
<td>is</td>
<td><img src="image" alt="Table 12 Illustration of Gaussian shape correspond with word in the caption sentence." /></td>
</tr>
<tr>
<td>playing</td>
<td><img src="image" alt="Table 12 Illustration of Gaussian shape correspond with word in the caption sentence." /></td>
</tr>
</tbody>
</table>
Example 2

Figure 28. Complex clip combined from 3 totally different clips.

Predicted sentences

Only clip 1 - A man is peeling a potato (GT - a girl is slicing a potato into pieces)
Only clip 2 - A cat is playing with a kitten (GT - a cat is cleaning itself)
Only clip 3 - A man is playing a guitar (GT - a man is playing a guitar)
Three clips combined - A man is playing a potato

Table 13 Illustration of Gaussian shape correspond with word in the caption sentence.
potato
8.5 Sample caption

This section provides predicted captions from videos in both MSVD and MSRVTT dataset, where caption quality varies from correct, incorrect but relevant, to totally incorrect.

8.5.1 MHB model

MHB: a man is cutting a piece of meat
GT: a person is slicing a rack of ribs

MHB: a woman is peeling a potato
GT: a woman is cutting the stems and leaves of an herb into pieces

MHB: a group of men are dancing in a field.
GT: a soccer player is kicking a soccer ball into the goal at a game

Figure 29 Sample captions for MHB models. From top to bottom: correct, incorrect but relevant, incorrect.
8.5.2 Baseline (GA-5) + HGA model

Baseline: a man is playing a guitar
GT: a man is playing a guitar

Baseline: a man is holding a piece of food
GT: a man is eating a slice of pizza

Baseline: a white animal is playing with a toy.
GT: two guinea pigs are eating leaves

Figure 30 Sample captions for baseline model (GA-5). From top to bottom: correct, incorrect but relevant, incorrect.
Figure 31 Sample captions for HGA models. From top to bottom: correct, incorrect but relevant, incorrect.

Baseline: a man is riding a motorcycle.
GT: a man is riding a motor bike in beach

Baseline: a car is driving down the street.
GT: a cop is trying to arrest a car driver.

Baseline: a polar bear is walking on the ice
GT: two polar bears are wrestling in the snow
Chapter 9  Conclusion and Future Work

We introduce a general purpose hierarchical framework for video understanding. Rather than use fixed training priors, we use video attributes as features along the length of the video to smartly steer the attention. When these temporal video features are bundled with a video summary vector, a semantically rich latent representation continuously feeds the captioning engine. A Gaussian parametric descriptor affords the ability to input variable length videos without padding or cropping. The usage of hierarchical recurrent models is both efficient and robust. We demonstrate state-of-the-art captioning results on multiple video datasets.

We also introduce a multi-stream captioning framework which can deal with both simple and complex videos. Our method uses a multi-stream hierarchical approach along with automatic boundary detection and parametric soft attention models to deliver state-of-the-art results on popular video captioning datasets.
REFERENCES


34. Sutskever, I., O. Vinyals, and Q.V. Le. Sequence to sequence learning with neural networks. in Advances in neural information processing systems. 2014.


on intrinsic and extrinsic evaluation measures for machine translation and/or summarization. 2005.


