1 Abstract

The solution to a considerable fraction of the problems that we aim to solve fall into the category of encoder-decoder based methods. We may wish to design exceedingly complex networks to solve sophisticated challenges like automatically describing an arbitrary image or translating a sentence from one language to another.

The neural encoder-decoder framework has recently been exploited to solve a wide variety of challenges in natural language processing, computer vision, speech processing, and even interdisciplinary problems. Some examples of problems that can be addressed by encoder-decoder based models are machine translation, automatic image and video caption generation, textual and visual question answering, and audio to text conversion.

The encoder part in this model is a neural structure that maps raw inputs to feature space and passes the extracted feature vector to the decoder. The decoder is also another neural structure that processes the extracted feature vector to make decisions or generate appropriate output based on the problem.

A wide variety of encoders are proposed to encode different types of inputs. Convolutional neural networks are typically used in encoding image and video inputs. Recurrent neural networks are widely used as encoders where the input is a sequence of structured data or sentence. In addition, more complex structures of different neural networks have been used to model complexities in inputs. Hierarchical CNN-RNN structures are examples of neural combinations which are widely used to represent temporal dependencies in videos which are used in video description generation.

Modeling long-term dependencies is an important issue that researchers should cope with while designing neural decoders. So a wide variety of techniques are used to model longer dependencies, such as trying to make deeper decoders by stacking RNNs, applying more nonlinearities, and modeling local and global temporal structures in a hierarchical manner.

Another potential issue with this baseline encoder–decoder approach is that the encoder has to compress all the necessary information of the input into a fixed-size tensor. This may make it difficult for the neural network to model temporal dependencies in both input and output. Attention mechanism is introduced to overcome the problem of fixed-length feature extraction as an extension to the encoder–decoder model. The distinguishing feature of this approach from the baseline encoder–decoder is that it does not attempt to encode a whole input into a single fixed-size tensor. Instead, it encodes the input into a sequence of annotation vectors and selects a combination of these vectors adaptively while decoding and generating the output in each

step.

The first section of this chapter introduces the baseline encoder-decoder model and its application in machine translation. Section 2, discusses different types of encoders and their applications in details and makes a general perspective of encoder structures in different problems. Section 3, provides a comprehensive study of decoder structures, techniques of making deeper decoders, along with their applications in image/video caption generation. Section 4, introduces the attention mechanism and its usage in machine translation, Following by an empirical study of the attention mechanism in other problems.

2 Baseline Encoder-Decoder Model

In this section, we aim to introduce the very baseline encoder-decoder model. In order to make a good understanding of the idea, the basic structure is represented to solve the machine translation task in which the model is designed to translate a sentence from the source language to the destination one.

2.1 Behind the idea and introduction

A wide range of problems in natural language processing, computer vision, speech recognition, and some multidisciplinary problems are solved by encoder-decoder based models. More specifically, the sophisticated problems, in which generating an output (which in the most cases is a sequential output such as a text) given an input is desired, can be solved by models based on encoder-decoder structure.

The main idea behind the framework is that the process of generating output can be divided into two subprocesses such that:

1. Encoding phase: First, the given input can be projected into another space by a projection function called "*encoder*" in order to make a "*good representation*" from the input. The encoder also can be viewed as a feature extractor from the input and the projection process can be expressed by a feature extraction process. In addition, the second space can be the "*feature space*" or the "*meaning space*" because the projected point represents the meaning of the input and all necessary information from the input that can help to generate the desired output.

2. Decoding phase: After encoding phase, a "*meaning vector*" is generated for the given input that well represents its meaning. In the second

phase, another projection function is required to map the meaning vector to the output space which is called "*Decoder*".

Figure 1 demonstrates the basic schema of the encoder-decoder framework. Let $X = \{X0, X1, X2\}$ denote the inputs and $Y = \{Y0, Y1, Y2\}$ denote the outputs of the problem, the decoder extracts a feature vector from the input and passes it to the decoder. The decoder then generates the output based on the extracted feature by the encoder.

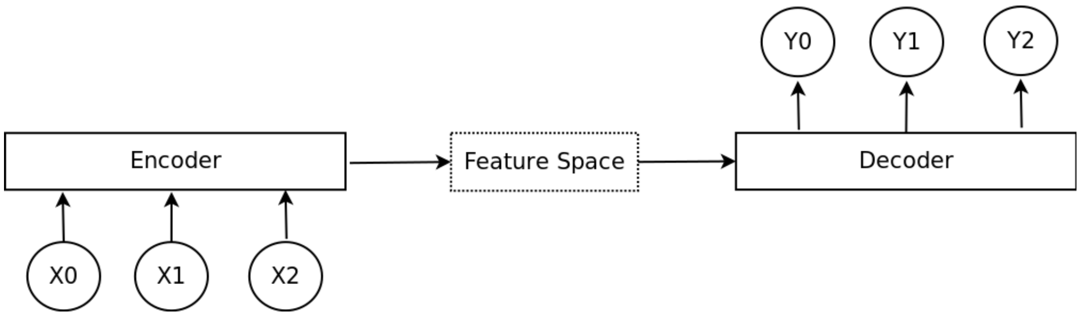

Figure 1: The basic scheme of the encoder-decoder model

2.2 Problems can be solved by encoder-decoder framework

Encoder-decoder framework is an architecture typically used to solve problems in which the model is supposed to generate a sequential output according to the given input from unconstrained type. In addition, the desired sequential output mostly is in the form of a text describing the input.

Some examples of the tasks in which the encoder-decoder model is used to solve the problem are listed bellow:

1. Machine Translation:
   "*Machine Translation*" (MT) is the task of generating a sentence from a destination language which has the same meaning of the given sentence from the source language. There exists two different approaches in machine translation.

   The first approach in MT is called "*Statistical Machine Translation*" (SMT). These kind of approaches are characterized by the use of statistical machine learning techniques in order to automatically translate the sentence from the source language to the destination. In less than two decays SMT has come to dominate academic Machine Translation research[1].

The second approach is called "*Neural Machine Translation*" (NMT). The encoder-decoder framework is first proposed in neural machine translation by Cho et al. in 2014 [2]. In the proposed model by cho et al. [2] a neural network is used to extract features from the input sentence and another neural network is used to generate a sentence word by word from the destination language with the same meaning of the input sentence using the extracted feature vector.

In the neural structures used in NMT, a neural network is trained to map the input sequence (the input sentence as a sequence of words) to the output sequence. This kind of learning is known as "*Sequence to Sequence Learning*".

Evaluations on the early models of NMT showed that however the generated translations are correct very well but the model faces extreme problems when translating long sentences[3]. The problem of modeling "*long term dependencies*" is one of the most important challenges in encoder-decoder models which we will drill into that and take a look at the proposed solutions for that, later in this chapter.

2. Image/Video Captioning:
   Image and Video Captioning are the problems of associating a textual description to a given image or video which holistically describes the presented objects and events in the input. There exists a wide variety of approaches including Probabilistic Graphical Models (PGMs) to Neural Encoder-Decoder based models proposed by researchers in these fields.

   Encoder-decoder based models for image captioning use a Convolutional Neural Network (CNN) as an encoder to extract feature vector from the input image and pass it to a Recurrent Neural Network (RNN) as decoder to generate caption. The model architecture in this task is the same as that of machine translation except that the encoder uses a CNN to encode image rather than using RNN.

   In addition in video captioning, also called "*video description generation*", a similar model based on encoder-decoder architecture is employed to generate a caption for the input video. In video captioning models the encoder typically consists of CNNs or combination of CNNs and RNNs to encode the input video and the decoder is the same as the decoder in machine translation and image captioning.

3. Textual/Visual Question Answering:
   Textual and Visual Question Answering are the problems of generating

an answer to a given question about an article and about an input image respectively. Proposed models to solve these problems are supposed to generate a short or long answer given a pair of an article and a question or an image and a question. The base model architecture is then similar to that of machine translation except that the encoder is required to extract a feature vector for a pair of inputs. The decoder is the same as the decoder in machine translation and image/video captioning because it is supposed to generate a sentence describing the meaning of the generated feature vector by encoder.

4. Text Summarization:
   Proposed models for summarizing a text are supposed to generate a textual summary for the input text. The only constraint on output is that it is required to describe the same meaning as the input text and its length should be shorter than that of the input. The base architecture of these models are the same as the architecture proposed in machine translation except that the generated output here is from the same language as the input.

It is clearly obvious that the baseline architecture proposed in machine translation is used in the other tasks and challenges with minor changes. In addition the decoder of the models in different tasks are somehow the same as each other because most of them are used to generate a sentence word by word to describe the meaning of the input represented by the feature vector extracted with the encoder. On the other side, a wide variety of encoders are used in order to extract appropriate feature vectors depending on the input types in different tasks.

2.3 The baseline encoder-decoder model for machine translation

As mentioned before, the machine translation is the problem in which encoder-decoder based models originated and first proposed. The Basic concepts of these models are shaped in machine translation literature. So, in this section we will introduce the basic encoder-decoder structure proposed in machine translation by cho et al. [2] to through a light over the model and its basics.

2.3.1 Formulation

Both the input and the output of machine translation models are sentences which can be formulated as a sequence of words. Let $X = \{x_0, x_1, \cdots, x_{L_i}\}$ denote the input sentence where x_i is the ith word in it assuming that the input sentence has L_i words. Similarly, the output sentence could be formulated as $Y = \{y_0, y_1, \cdots, y_{Lo}\}$ in which y_i is the ith word in the output sentence assuming that the output sentence has L_o words. Furthermore, all of x_is and y_is are one-hot vectors created from a dictionary of all words in the datasets.

In addition, sentences can be modeled using *Bag of Words* (BoW) technique in which the presence of a word in the sentence is considered without any information about the order of words. Furthermore, all of the words in the input and the output sentences can be represented by *one-hot* vectors. A one-hot vector is a vector that all of its components are zero except one of them. In order to create a one-hot vector for each word, first a dictionary[1] of all possible words in the available datasets is created. Assuming N words in the dictionary, an N-dimensional zero vector for each word is created and the component with the same index of the word in the dictionary is set to one. Figure 2 demonstrates one-hot vector for each word in a sample dictionary. Assuming the dictionary D has 5 words "I", "cat", "dog", "have", "a" sequentially with the indices 0 to 4, one-hot vector for each word is displayed in the figure.

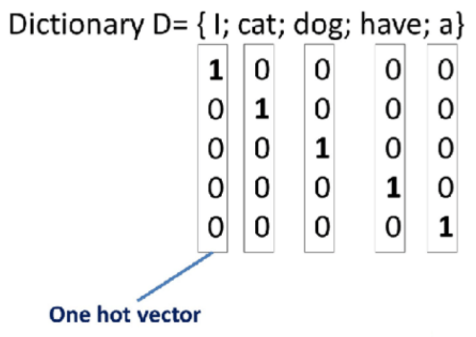

Figure 2: Appropriate one-hot vector for each word in a sample dictionary

The translation process is divided into two subprocesses as bellow:

1. Encoding phase:
   A Recurrent Neural Network (RNN) is used to extract a feature vector from the input sentence from the source language. All of the words in the input sentence are converted to one-hot vectors and passed to the RNN in the order of their presence in the sentence. The RNN then updates its hidden state and output vectors according to each

[1]A dictionary is a list of unique words with unique indexes

word. The iteration is stoped when the *End of Sentence* (EOS) token is passed to the RNN. The EOS token is a token added to the end of input setences manually to determine the end point in the sentences. The hidden state of the RNN after the EOS token is then used as feature vector of the input sentence. One-hot vectors of the words in input sentence are created using the dictionary of words from the source language.

2. Decoding phase:
   Another RNN is used to generate words of the output sentence in order. The decoder RNN is designed to predict a probability distribution over all possible words in the dictionary of source language words at each step. Then a word is selected w.r.t. the produced probability distribution as the next word in the sentence. The iteration is stopped when the EOS token is generated by the decoder or a predefined number of words are generated.

The whole structure of the proposed model is demonstrated in figure 3. The context vector extracted by the encoder is denoted by C which is the hidden state of the RNN encoder at the last step.

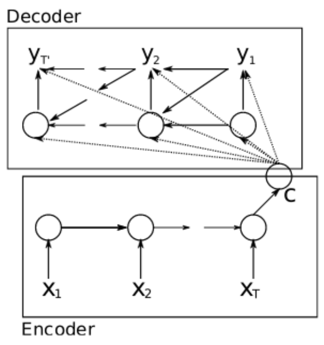

Figure 3: An illustration of the first encoder-decoder based model proposed by cho et al. [2]

2.3.2 Encoder Structure

An RNN is used as the encoder in the proposed model by cho et al.[2]. Let h_e denote the hidden state of the encoder RNN. The hidden state of the encoder is updated at each time step t according to equation (1) in which h_e^t is the hidden state of the encoder at time step t, $f_{encoder}$ is a nonlinear activation function that can be as simple as an element wise logistic sigmoid function and as complex as a Long Short-Term Memory (LSTM), and x_t is the one-hot vector of the tth word in the input sentence.

$$h_e^t = f_{encoder}(h_e^{t-1}, x_t) \tag{1}$$

Assuming that the input sentence has L_i words, the encoder RNN should iterate on each word and update its hidden state vector at each step. The hidden state of the RNN after the L_ith word is then passed to the decoder as the context vector C. So, the context vector extracted by encoder can be computed as equation (2).

$$C = h_e^{L_i} \tag{2}$$

2.3.3 Decoder (language modeling)

The decoder is supposed to generate the output sentence word by word in a way that the meaning of the sentence is the same as the meaning of the input sentence represented by context vector C. From another point of view, the decoder can be seen as an RNN that maximizes the likelihood of the translated sentence in the dataset for the input sentence and its generated context vector as expressed in (3) in which θ is the set of all trainable weights and parameters of the model.

$$Pr_\theta\{Y|X\} \tag{3}$$

On the other side, according to the encoder-decoder structure, the random variables C depends directly to the random variable X, and the random varialbe Y depends directyl on random variable C. Figure 4 displays the dependency graph between these 3 random variables.

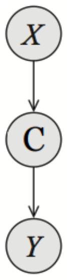

Figure 4: Directed Acyclic Graph of dependecies of random variables in encoder-decoder model

According to the dependencies displayed in the figure 4, it can be derive that $Y \perp\!\!\!\perp X|C$. Since given C we can replace the likelihood expressed in (3) with the likelihood expressed in (4). Furthermore, assuming that each word in the sentence depends only on the meaning of the previous words in the sentence the probability of a sentence can be replaced by the multiplication of the probability of each of its words given previous ones.

$$Pr_\theta\{Y|C\} = \Pi_{t=0}^{L_o} Pr\{y_t|y_{t-1}, y_{t-2}, \cdots, y_0, C\} \tag{4}$$

Consequently the decoder is supposed to generate a probability distribution over each word at each step t given its previously generated words and the extracted context vector by the encoder. We can formulate the probability distribution by the RNN according to equations (6) and (7). Let h_d^t be the hidden state of the decoder at time step t. Let O^t be a L_o dimensional vector and the output of the decoder at time step t which is generated by a nonlinear function g on the decoder's hidden state and the context vector C. The decoder's hidden state also is generated by the nonlinear function $f_{decoder}$ on the previously generated word, hidden state of the decoder at previous time step and the context vector (5) With applying a softmax on the output, a vector with the same size is generated which sum of its components is equal to one and can be treated as the desired probability distribution.

$$h_d^t = f_{decoder}(h_d^{t-1}, y_{t-1}, C) \tag{5}$$
$$O^t = g(h_d^t, y_{t-1}, C) \tag{6}$$
$$Pr\{y_t|y_{t-1}, y_{t-2}, \cdots, y_0, C\} = SoftMax(O^t) \tag{7}$$

At each time step, the probability distribution $Pr\{y_t|y_{t-1}, y_{t-2}, \cdots, y_0, C\}$ is generated by decoder according to equation (7) and the next word is selected w.r.t. this probability distribution over the words in the dictionary of the destination language.

The two components of the proposed model can be jointly trained to minimizing the negative conditional log likelihood expressed by (8) in which N is the number of samples in the dataset, Y_n and X_n are the nth output and input pair in the dataset, θ is the set of all trainable parameters, and $Loss$ is the loss function to be minimized.

$$Loss = -\frac{1}{N}\Sigma_{n=0}^{N} log Pr_\theta(Y_n|X_n) \tag{8}$$

3 Different Encoders and Their Applications

The baseline encoder-decoder architecture proposed by cho et al. [2] in machine translation, attracted many researchers attention in different fields. As explained before, almost all of the variants of the baseline architecture in different tasks share a similar decoder but the structure of encoder varies with according to the type of inputs. In this section we will introduce the important structures of encoders to encode different input types.

3.1 Sentence as Input

In fact, the most simple encoder for the problems with sentences as inputs is an RNN. The first proposed encoder in machine translation is an LSTM which takes all words of the input sentence, processes them and returns its hidden state vector at the end of the sentence as the context vector.

Along with the RNNs, CNNs are employed to extract features from the source sentences in encoding phase. As an instance, Gehring et al. proposed a convolutional encoder for machine translation in order to create better context vectors by taking nearby words into consideration using a CNN[4]. In the proposed encoder by Gehring et al.[4], a CNN with a kernel size $k = 3$ is used to extract a combination of each three nearby words' meaning in the sentence to generate context vector.

In addition, different RNN cells are used as blocks of encoder for sentence inputs. LSTMs[5] are widely used because of their ability to cope with the long term dependencies and remembering far history in the input sequence [6][7][8][9]. GRU [2] is also used in different proposed models because of its good performance and that it can be assumed as a light-weighted version of LSTM [2][10][11][12].

3.2 Image as Input

Encoder-Decoder based architectures form a majority of proposed models to generate captions for images. In such models, the process of generating captions for the input image is divided into two steps. The first step is encoding in which a feature vector extracted from the image is returned as context vector. The second step is decoding in which the generated context vector is passed to a decoder to generate sentences describing the context.

The best choice for encoders in such problems is Convolutional Neural Network (CNN). Almost all of the proposed models for image captioning based on encoder-decoder framework are using different types of CNNs as encoders. Neural encoder-decoder based approaches to image captioning share the same structure for decoder while in most of them the encoder consists of a single CNN. So, the extracted feature vector from the image can be expressed by the equation 9 in which X is the input image, $CNN(X)$ is the output of the CNN network, and C is the context vector passed to the decoder.

$$C = CNN(X) \tag{9}$$

A wide variety of CNNs are employed as encoders in the proposed models for image captioning. Since, pretrained version of VGGNet [13] and AlexNet [14] on ImageNet dataset [15] are available online and extract good features from images for different tasks, they are used as encoders in different proposed image captioning models [16] [17] [18]. Furthermore, ResNet[19] is widely used because of its good performance as the encoder in such models [20] [21] [22] [23]. Google NIC Inception v3 [24] is also used in proposed models because of its better image classification accuracy rather than ResNet [25] [26] [27] [28]. Figure 5 illustrates the architecture of encoder-decoder based models in image captioning.

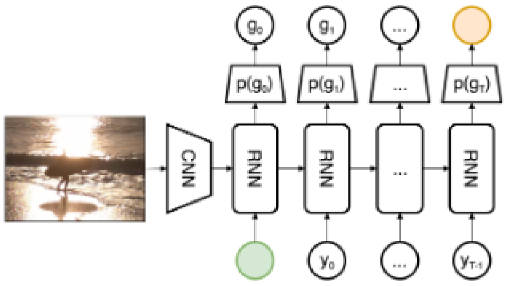

Figure 5: Model architecture based on encoder-decoder framework for image captioning [27]

3.3 Video as Input

Another sort of problems that encoder-decoder models play an important role in solutions proposed for them are those with videos as the input and describing text as the output, also called "*Video Description Generation*" or "*Video Captioning*". Since, creating a good representation is critical to overall performance of video captioning models, a wide variety of encoders to cope with different difficulties and challenges of videos as inputs are proposed. In this section some examples of encoders to deal with the challenges of extracting motion details from the video are pointed out.

Assuming an input video V consists of L_i frames. We can formulate the input video as (10) in which v_i is a representation of the ith frame in the input video and v_{L_i} is the end of video token (<EOV>). In fact, each v_i is the feature vector extracted by a CNN on the ith frame in the input video.

$$V = \{v_0, v_1, \cdots, v_{L_i}\} \tag{10}$$

Since, in the baseline encoder-decode model, encoder should return a "*fixed length*" context vector extracted from the input, an aggregation function is required to aggregate feature vectors from different frames in the video and pass it as the context vector to the decoder.

Different ideas are employed to propose a good aggregation for video captioning. The first end to end encoder-decoder based approach in video description generation proposed by venugopalan et al. in 2014 [29] used a mean pooling layer to create the fixed length context vector from the input video. In the proposed model by venugopalan et al. [29], first a CNN is applied to each frame of the input video. Then a mean pooling layer is applied to create an average feature vector over the set of feature vectors extracted from each frame. The average feature vector is then passed to the decoder to generate the sentence. A stacked RNN structure is used in decoder which we will drill into it later.

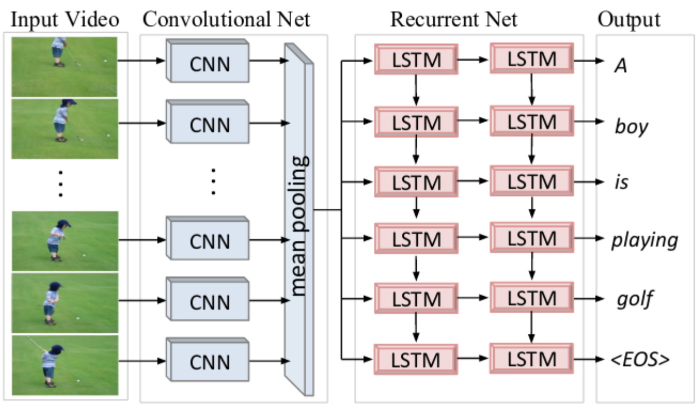

Figure 6: An illustration of the first encoder-decoder based model for video description generation proposed by venugopalan et al. in 2015 [29]

Different CNNs are used to extract feature vectors from frames of the input video. For instance, Yao et al. [23] is used Inception V4 [30] for feature extraction from video frames.

Extracting good features from the input video is a challenging task that can extremely affect the performance of proposed model. The extracted context vector from the input video should well express the detailed motions in the video. In order to create an encoder which is able to extract fine motion features from the video, Yao et al. [31] proposed an encoder-decoder based approach that uses a 3D-CNN as the encoder. The structure of proposed 3D-CNN is illustrated in figure 7.

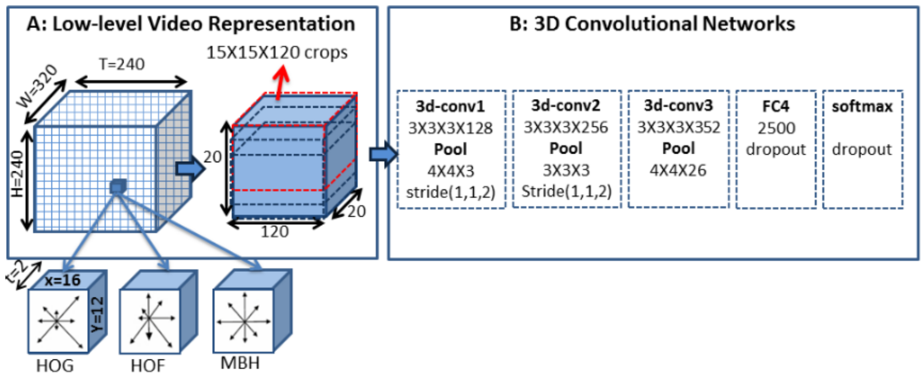

Figure 7: The structure of 3D-CNN proposed by Yao et al. in 2015 [31]

Actually, the proposed 3D-CNN models spatio-temporal dependencies in the input video. The 3D-CNN is used to build the higher-level representation that preserves the local motion information from short frame sequences in the input video. This is accomplished by first dividing the input video clip into a 3D spatio-temporal grid of $16 * 12 * 2$ (width * height * timesteps) cuboids. Each cuboid is represented by concatenating the histogram of oriented gradients (HoG), histogram of oriented flow (HoF) and motion boundary histogram(HoG, HoF, MbH) with 33 bins. This transformation ensures that local temporal structures and motion features are well extracted. The generated 3D descriptor then is passed to 3 convolutional layers each following by a max-pooling layer and one fully connected layer followed by a softmax layer as demonstrated in the figure 7. The output of the 3D-CNN is then passed to the decoder to generate appropriate caption.

The proposed 3D-CNN by Yao et al. [31] is also used along side the typical 2D-CNN in other works. Pan et al. [32] proposed a novel encoder-decoder architecture for video description generation and used the 3D-CNN and the typical 2D-CNN and applied a mean pooling layer to the set of extracted features by each of the CNNs and concatenated the output to generate the context vector of the video. Figure 8 illustrates the encoder part of the proposed model by Pan et al. [32].

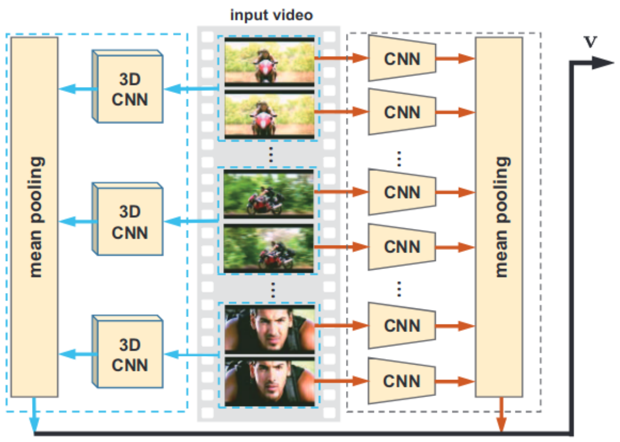

Figure 8: An illustration of the encoder part of the proposed model by Pan et al. [32]

Another kind of approaches to video captioning are those focusing on "*Dense Video Captioning*". Dense video captioning is the task of video captioning in which the proposed model is able to first localize displaying events in the input video and then generate a description for each of the localized events.

Encoders for dense video captioning task are supposed to first detect all of the displaying events in the input video. Then for each of the events a quadruple $<t_{start}, t_{end}, score, h>$ should be extracted. t_{start} and t_{end} are the number of starting and ending frames of the specified event. $score$ is the confidence score of the encoder for each of events. If the $score$ of an event is greater than a threshold, it is reported as an event and its quadruple is passed to the decoder for sentence generation, otherwise it is ignored. Finally h is the feature vector extracted from the range of frames between t_{start} and t_{end} which is used by decoder as context vector of the event to generate a sentence for that[33].

The task of dense video captioning is proposed by Krishna et al. [34] first in 2017. The proposed encoder by Krishna et al. [34] for dense video captioning task is able to identify events of the input video within a single pass while the proposed decoder simultaneously generates captions for each event detected and passed by the encoder.

Figure 9 illustrates the structure of proposed encoder by Krishna et al. [34] for dense video captioning. The proposed encoder is able to extract all events in the input video using a deep action proposal(DAP) module proposed by [35]. To do this, a 3D-CNN is applied to the input video frames and extracts video features. The extracted video features are passed to the

DAP module. DAP module consists of different LSTMs that are applied to the video features sequence in different resolutions and are trained to detect starting and ending points of events. The confidence score of each event is also computed by DAP. the proposed event proposals are then sorted w.r.t. their ending points and passed sequentially to the decoder. The feature vector of each event is also the hidden state of the corresponding RNN in the DAP. The decoder then generates a sentence for each event using its feature vector as the encoder output.

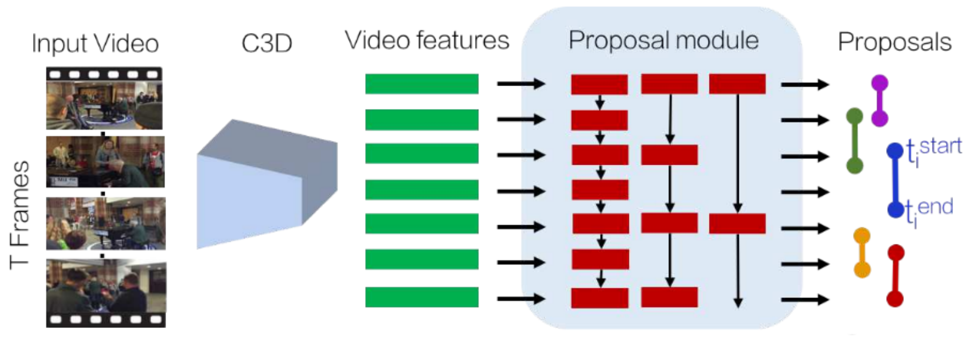

Figure 9: An illustration of the first proposed encoder model for dense video captioning by Krishna et al. [34]

Li et al proposed a novel end to end encoder-decoder based approach to dense video captioning which unified the temporal localization of event proposals and sentence generation[33]. Figure 10 illustrates the structure of proposed model by Li et al. [33]. In the proposed model, instead of using an extra DAP module, a 12 layer convolutional structure is designed to extract features for action proposal over the output of the 3D-CNN. The first 3 layers of the convolutional structure (500D layer and base layers in the figure 10) are designed to make nonlinearities and decrease the input dimension. The next 9 layers, which are called "Anchor layers", extract features from different resolutions to be used for event prediction. The "Prediction Layer" consists of 3 parallel fully connected layers to first regress temporal coordinates (t_{start} and t_{end}) of each event, second compute the descriptiveness of the event ($score$) and third to classify event vs background. The prediction layer is applied to the output of all anchor layers to make model able to detect events from different resolutions. The extracted proposals are then passed to the proposal ranking module which ranks event proposals w.r.t. their ending time and finally events are passed to the decoder for sentence generation sequentially.

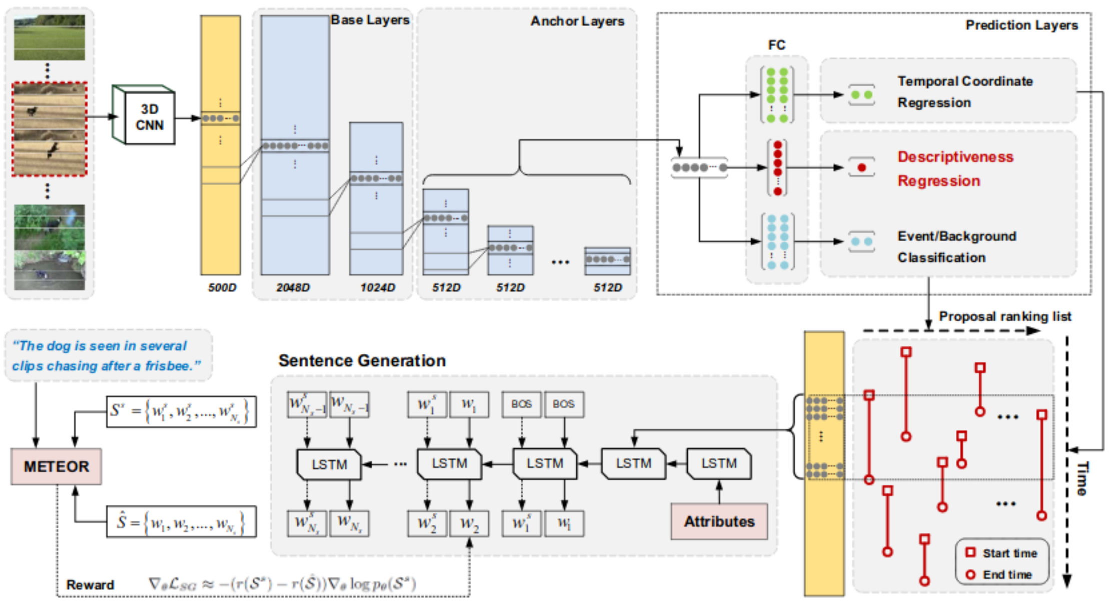

Figure 10: An illustration of the encoder-decoder structure proposed by Li et al. for dense video captioning [33]

A wide variety of models are proposed to cope with the difficulties of encoding phase in dense video captioning. Shen et al. [36] proposed a new CNN called "*Lexical FCN*" which is trained in a weakly supervised manner to detect events based on the captions in the dataset. Duan et al. also proposed a novel approach for dense video captioning based on the similar assumption "each caption describes one temporal segment, and each temporal segment has one caption"[37]. Xu et al. proposed an end to end encoder-decoder based model for dense video captioning which detects and describes events in the input video jointly and is applicable for dense video captioning on video streams. Zhou et al. also proposed an end to end approach with a masking network to localize and describe events jointly[38]. Wang et al. proposed a novel architecture to take both past and future frames into account while localizing the events in the input video [39] using bidirectional models.

4 Decoder Structures

As it is explained before, decoders in encoder-decoder based models generate a sequential output for the given input. The output is mostly in the form of a descriptive text. Therefore, the main structure of the decoders are similar to each other in different tasks. In this section, different techniques to make better decoders with better generated captions are studied.

4.1 Long term dependencies

One of the basic problems with the RNNs is the problem of "*long term dependencies*". Indeed, in this sort of neural networks when the length of the input or the length of the desired output is too large, the gradients should propagate over many stages. When the gradient is propagated over a large number of stages, they tend to either vanish or explode.

In addition, the gradients in each backpropagation step are multiplied by small coefficients like learning rates. Thus, the gradient in the early stages will be close to zero and might make no changes in weights of the early stage layers[40].

In this section we will study approaches to cope with the long term dependencies challenge in decoders among proposed models based on encoder-decoder framework in different tasks.

4.2 Using LSTMs

LSTMs obtained strong results on a variety of sequence modeling tasks thanks to their superior ability to preserve sequence information over time. The combination of the "*memory cell*" and the "*forget gate*" in the structure of LSTM improves the ability of LSTM to model sequence information by training to forget unnecessary information (using forget gate) and keep necessary information in memory cell. Cho et al. [2], Bahdanau et al.[7], Luong et al. [41], Wu et al. [42], Johnson et al. [43] and Luong et al. [44] used LSTMs as both the encoder and decoder part of their proposed models for machine translation because of the good power to remember far points.

4.3 Stacked RNN

As explained earlier, multi-staged decoders are hard to train due to the vanishing gradient problem. Thus most of the proposed encoder-decoder based models use a single layer RNN as the decoder which results in difficulties to generate rich fine-grained sentences. Stacking multiple RNNs on top of each other, is another way to enable decoders to generate sentences describing more details of the input image.

Donahue et al. [45] proposed an encoder-decoder based approach to image captioning which uses a stacked structure of LSTMs as decoder in order to describe more details of the input image. In the proposed method by Donahue et al. [45] an LSTM is used on top of another one in the way that the first layer LSTM takes image features and the previously generated word embedding along with its previous hidden state vector as input and generates a coarse low-level representation of the output sentence. In the next step, the hidden state of the low level LSTM is passed to the next LSTM as input alongside its previous hidden state to generate fine high-level representation of the output. A softmax layer is then applied to the generated high-level representation of the output to generate the probability distribution of the next word in the sentence. Gu et al. [46] also used en encoder-decoder based model with a two-layer stacked LSTM as decoder in order to enable the proposed model to generate better descriptions.

The idea of employing a stacked structure of RNNs as decoder is also used in proposed models for video description generation. Venugopalan et al. [29] proposed the first decoder in neural encoder-decoder based approaches to video description generation with a simple stacked structure. Figure 11 demonstrates the architecture of a sample stacked decoder. The red line illustrates the shortest path from the first step to the output in the model. Furthermore, blocks tagged by ”C” display the input in each step.

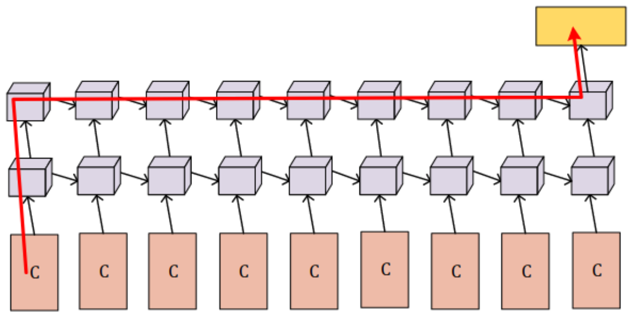

Figure 11: Stacked structure of RNNs [47]

Along with the methods using stacked RNNs as decoders, a category of models are proposed with following a hierarchical fashion to arrange RNNs in decoders in order to enable models to generate fine-grained output sequences.

In addition, hierarchical RNN structures are furthermore used to enable encoders in the problems with a sequential input to exploit and encode more detailed information from the input. Pan et al. [47] proposed an encoder-decoder based model for video description generation with a hierarchical encoder structure. In their proposed model, two layers of different LSTMs are used. The first layer LSTM is applied to all sequence steps in order to exploit low-level features and the second layer LSTM is applied on the output of equally sized subsets of the input sequence steps to exploit the high-level features. Figure 12 demonstrates the architecture of the proposed hierarchical structure by Pan et al. [47]. The illustrated red line, denotes the shortest path from the first step to the output. Comparing structures displayed in figures 11 and 12 shows that the shortest path from the first step to the output in hierarchical models is way smaller than that of stacked models. Therefore, the efficiency of the hierarchical model is much better than stacked model.

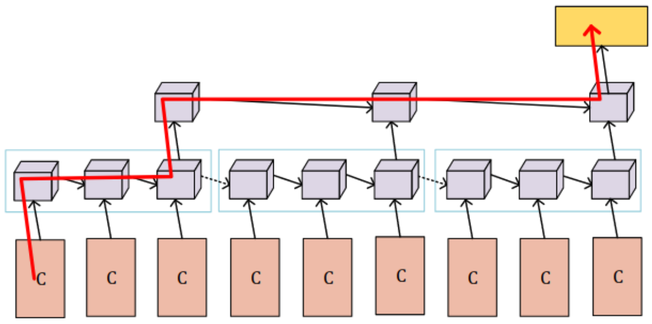

Figure 12: Hierarchical structure of RNNs [47]

Furthermore, more complex hierarchical structures are proposed with different intents in the literature. Yu et al.[48] proposed a model with a hierarchical structure to generate a set of sentences arranged in a single paragraph as a description for the input video. The first layer in this model is a simple decoder to generate single sentences and the first layer is a "*paragraph controller*". The paragraph controller is another RNN which generates a feature vector given the last hidden state of the first layer RNN denoting the meaning of the next sentence to be generated. The first layer RNN is then take the feature vector generated by the second layer and concatenates it with other inputs to control the meaning of the new generating sentence.

4.4 Reinforcement learning

One of the problems of training the decoders using loglikelihood model is that the performance of the model is highly different on the training and testing sets. It occurs since the optimization function for training is different from the evaluation metrics while testing. Recently, reinforcement learning is used to decrease the gap between training and testing performance of the proposed models. In other words, the main problem with loglikelihood objective is that it does not reflect the task reward function as measured by the BLEU score in translation.

Wu et al. [42] proposed the first decoder trained by reinforcement learning for machine translation. After that, other researchers used reinforcement learning to train proposed decoders in other tasks. Wang et al. [49] proposed the first decoder trained with reinforcement learning taking CIDEr[50] score as reward in video captioning task. Li et al. [33] also trained proposed decoder in reinforcement learning fashion using METEOR[51] score as reward to generate caption for the input videos.

Figure 13 illustrates the structure of the proposed model by Wang et al. [49]. The proposed decoder, consists of three different modules manager, worker, and internal critic. All of the three modules proposed in the decoder are trained using a reinforcement learning fashion. The manager operates at a lower temporal resolution and emits a goal when needed for the worker to accomplish, and the worker generates a word for each time step by following the goal proposed by the manager. The internal critic determines if the worker has accomplished the goal and sends a binary segment signal to the manager to help it update goals.

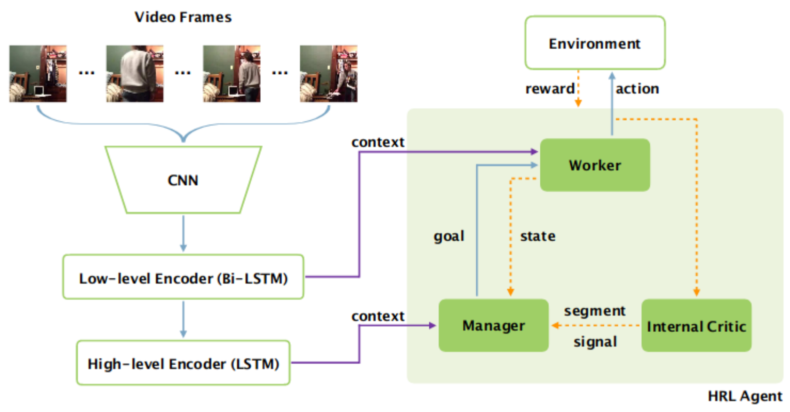

Figure 13: An illustration of the encoder-decoder based model proposed by Wang et al. [49]. The proposed decoder is trained using reinforcement learning.

Figure 14 illustrates the unrolled decoder proposed by Wang et al. [49]. The manager takes the context vector c_t^M at time step t and the sentence step at previous time step h_{t-1}^W as input and uses an LSTM to the semantic goal according to the equations (11) and (12).

$$h_t^M = LSTM^M(h_{t-1}^M, [c_t^M, h_{t-1}^W]) \tag{11}$$
$$g_t = u_M(h_t^M) \tag{12}$$

In the equations (11) and (12), $LSTM^M$ denotes the LSTM function used in manager, u_m is the function projecting hidden states to the semantic goal, h_{t-1}^M is the hidden state of the manager LSTM at previous time step, and g_t is the vector of semantic goal generated at time step t.

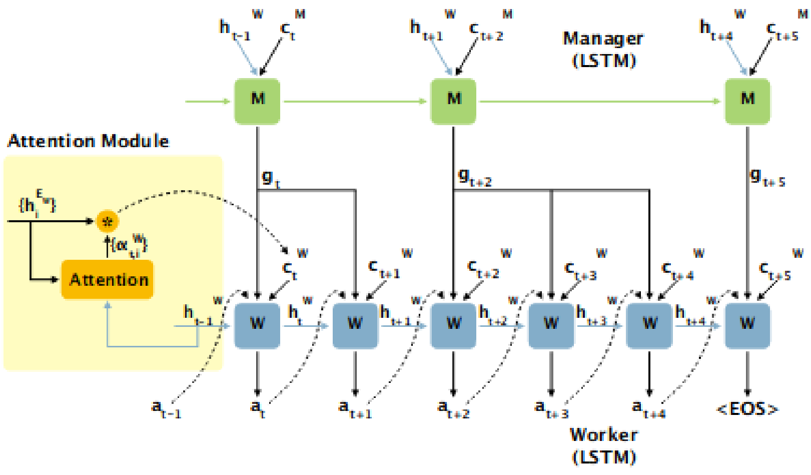

Figure 14: An illustration of the unrolled decoder proposed by Wang et al. [49].

The worker then receives the generated goal g_t, takes the concatenation of $[c_t^W, g_t, \alpha_{t-1}]$ as the input, and outputs the probabilities π_t over all actions $\alpha_t \in V$, where each action is a generated word according to equations (13) to (15).

$$h_t^W = LSTM^W(h_{t-1}^W, [c_t^W, g_t, \alpha_{t-1}]) \tag{13}$$
$$x_t = u_W(h_t^W) \tag{14}$$
$$\pi_t = SoftMax(x_t) \tag{15}$$

The internal critic is used to make a good coordination between manager and worker. Internal critic is indeed a classifier to determine when the worker is done with generating appropriate phrase for a given goal. When worker is done, internal critic sends an activation signal to manager to generate a new goal. Let z_t be the binary signal of the internal critic, the probability $Pr(z_t)$ is computed according to equations (16) and (17).

$$h_t^I = LSTM^I(h_{t-1}^I, \alpha_t]) \tag{16}$$
$$Pr(z_t) = sigmoid(W_z h_t^I + b_z) \tag{17}$$
$$\tag{18}$$

The objective of the worker is to maximize the discounted return (19) in which θ_W is the set of trainable parameters of the worker, γ is the discount rate, and r_{t+k} is the reward at step $t+k$. Therefore, the loss function of the decoder can be written as (20).

$$R_t = \Sigma_{k=0}^{\infty} \gamma^k r_{t+k} \tag{19}$$
$$L(\theta_W) = -E_{\alpha_t \sim \pi_{\theta_W}}[R(\alpha_t)] \tag{20}$$

he gradient of non-differentiable, reward-based loss function can be derived as:

$$\nabla_{\theta_W} L(\theta_W) = -E_{\alpha_t \sim \pi_{\theta_W}}[R(\alpha_t \nabla_{\theta_W} log \pi_{\theta_W}(\alpha_t))] \tag{21}$$

Typically the expectation of the loss function if estimated with a single sample, so the expectation term can be omitted. In addition, the reward can be subtracted with a baseline b_t^W in order to generalize the policy gradient.

$$\nabla_{\theta_W} L(\theta_W) \approx -(R(\alpha_t - b_t^W) \nabla_{\theta_W} log \pi_{\theta_W}(\alpha_t) \tag{22}$$

The manager is supposed to be trained in a way that can compute goals to generate sentences with better BLEU scores and the action of the decoder is produced by worker. So, the worker is assumed to be fully trained and used as a black box when training the manager. More specifically, the manager outputs a goal g_t at step t and the worker then runs c steps tp generate the expected segment $e_{t,c} = \alpha_t \alpha_{t+1} \alpha_{t+2} \cdots \alpha_{t+c}$ using the goal, then the environment responds with a new state s_{t+c} and reward $r(e_t, c)$. Following a similar math, final gradients to train manager can be computed as:

$$\nabla_{\theta_M} L(\theta_M) = -(R(e_t, c) - b_t^M)[\Sigma_{i=t}^{t+c-1} \nabla_{g_t} log \pi(\alpha_i)] \nabla_{\theta_m} \mu_{\theta_M}(s_t) \tag{23}$$

Which $\mu_{\theta_M}(s_t)$ is a noisy version of the generated goal and used in order to empower exploration in the training of the model. Furthermore, rewards are defined as (24) and (25).

$$R(a_t) = \Sigma_{k=0}^{\infty}\gamma^k[CIDEr(sent + \alpha_{t+k}) - CIDEr(sent)] \tag{24}$$
$$R(e_t) = \Sigma_{n=0}^{\infty}\gamma^n[CIDEr(sent + e_{t+n}) - CIDEr(sent)] \tag{25}$$

Furthermore, other metrics like BLEU score[52] can be used instead of CIDEr.

5 Attention Mechanism

The models based on encoder-decoder framework, encode input to a "*fixed length vector*" and the decoder in these models is supposed to generate the output based on information represented in the fixed length output of the encoder. On the other hand, each element in the output may be more related to a specific part of the input. In these cases, more detailed information about that specific part of the input is required while extra information from the other parts of the input could deceive the model.

Attention mechanism, first introduce by Bahdanau et al. [7] in machine translation, is a mechanism that provides encoder-decoder models to pay more attention to a specific part of the input while generating output at each step. Furthermore, the mechanism enables decoders to cope with the long term dependencies and generate more fine-detailed sentences and outputs.

In this section, first the basic idea of the attention mechanism proposed in machine translation is described. Then, the applications of the attention mechanism in proposed models based on encoder-decoder architecture in other tasks are studied.

5.1 Basic Attention Mechanism

Bahdanau et al. [7] proposed the first encoder-decoder based model equipped with the attention mechanism in order to make better translations. Both the encoder and the decoder of the proposed model are changed. The encoder is changed in a way that generates a sequence of feature vectors called "*annotation vectors*" and an extra layer called "*attention layer*" is augmented in between the encoder and the decoder. The attention layer, receives the annotation vectors generated by the encoder and creates a fixed length context vector at each step and passes it to the decoder in order to generate the probability of the next word in the sentence.

According to the described changes, the target probability distribution of the decoder can be expressed as (26) in which the probability of the next word at time step t, denoted by y_t given all previously generated words and the context vector generated to predict the tth word is computed by the decoder at each step.

$$Pr(y_t|y_{t-1}, \cdots, y_0, C_t) \tag{26}$$

Let $L = \{l_0, l_1, \cdots, l_{N_i}\}$ be the set of generated annotation vectors by the encoder and N_i be the number of generated annotations, the context vector C_t is then computed at each step following the equation (27). The multipliers α_k in (27) are called "*attention weights*".

$$C_t = \Sigma_{k=0}^{N_i} \alpha_k^t l_k \tag{27}$$

The keypoint in generating the context vector at each step using the attention mechanism is to compute the attention weights at each step. Many researchers have proposed different ways to compute attention weights. One of the most used attention mechanism is proposed by Xu et al. [9] which is called "*Soft Attention*". The attention weights in soft attention fashion are computed according to equations (28) to (29).

$$\alpha_k^t = \frac{exp(e_k^t)}{\Sigma_{j=0}^{N_i} exp(e_j^t)} \tag{28}$$

$$e_k^t = f(h_{t-1}, l_j) \tag{29}$$

The (29) is an alignment model which scores how well the output at step t is depends to the input section related to the annotation vector l_j. The function f in (29) measures the alignment between the output and the input. One of the simple candidates to implement function f is an MLP which can be modeled as:

$$f(h_{t-1}, l_j) = W_2 tanh(W_h h_{t-1} + W_l l_j + b_1) + b_2 \tag{30}$$

Which W_2, W_h, and W_l are weight matrices and b_1 and b_2 are added biases and all of these parameters can be trained jointly with other model trainable parameters.

Another version of the attention mechanism is also introduced by Xu et al. [9] in which at each step one of the attention weights is equal to 1 and the rest are equal to zero, called "*Hard Attention*". By the way, the proposed soft attention is widely used in other proposed models in different tasks.

Attention mechanism is also used in other tasks. You et al. [53] proposed a semantic attention in image captioning. Lu et al. [20] also proposed an adaptive version of the attention mechanism in image captioning. Gao et al. proposed an encoder-decoder based model using the attention mechanism and introduced a novel approach to train attention weights to decrease the semantic gap of the generated caption. Wu et al. [54] also proposed a novel encoder-decoder based model for video captioning which is able to select from different information sources like motion, temporal, audio and semantic features of the input video to generate caption describing the input video.

References

[1] A. Lopez, "Statistical machine translation," *ACM Computing Surveys (CSUR)*, vol. 40, no. 3, p. 8, 2008.

[2] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[3] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[4] J. Gehring, M. Auli, D. Grangier, and Y. N. Dauphin, "A convolutional encoder model for neural machine translation," *arXiv preprint arXiv:1611.02344*, 2016.

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

[6] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, pp. 3104–3112, 2014.

[7] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[8] M.-T. Luong and C. D. Manning, "Stanford neural machine translation systems for spoken language domains," in *Proceedings of the International Workshop on Spoken Language Translation*, pp. 76–79, 2015.

[9] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning*, pp. 2048–2057, 2015.

[10] H. Mi, B. Sankaran, Z. Wang, and A. Ittycheriah, "Coverage embedding models for neural machine translation," *arXiv preprint arXiv:1605.03148*, 2016.

[11] D. He, Y. Xia, T. Qin, L. Wang, N. Yu, T.-Y. Liu, and W.-Y. Ma, "Dual learning for machine translation," in *Advances in Neural Information Processing Systems*, pp. 820–828, 2016.

[12] Z. Tu, Y. Liu, Z. Lu, X. Liu, and H. Li, "Context gates for neural machine translation," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 87–99, 2017.

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

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255, Ieee, 2009.

[16] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3128–3137, 2015.

[17] L. Chen, H. Zhang, J. Xiao, L. Nie, J. Shao, W. Liu, and T.-S. Chua, "Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6298–6306, IEEE, 2017.

[18] M. Pedersoli, T. Lucas, C. Schmid, and J. Verbeek, "Areas of attention for image captioning," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1242–1250, 2017.

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

[20] J. Lu, C. Xiong, D. Parikh, and R. Socher, "Knowing when to look: Adaptive attention via a visual sentinel for image captioning," in *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 3242–3250, IEEE, 2017.

[21] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel, “Self-critical sequence training for image captioning,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7008–7024, 2017.

[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 vqa,” *arXiv preprint arXiv:1707.07998*, 2017.

[23] T. Yao, Y. Pan, Y. Li, Z. Qiu, and T. Mei, “Boosting image captioning with attributes,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 4904–4912, IEEE, 2017.

[24] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[25] L. Zhang, F. Sung, F. Liu, T. Xiang, S. Gong, Y. Yang, and T. M. Hospedales, “Actor-critic sequence training for image captioning,” *arXiv preprint arXiv:1706.09601*, 2017.

[26] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv preprint arXiv:1502.03167*, 2015.

[27] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, “Show and tell: A neural image caption generator,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3156–3164, 2015.

[28] S. Liu, Z. Zhu, N. Ye, S. Guadarrama, and K. Murphy, “Improved image captioning via policy gradient optimization of spider,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 873–881, IEEE, 2017.

[29] S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney, and K. Saenko, “Translating videos to natural language using deep recurrent neural networks,” *arXiv preprint arXiv:1412.4729*, 2014.

[30] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning.,” in *AAAI*, vol. 4, p. 12, 2017.

[31] 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 IEEE international conference on computer vision*, pp. 4507–4515, 2015.

[32] 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*, pp. 4594–4602, 2016.

[33] Y. Li, T. Yao, Y. Pan, H. Chao, and T. Mei, "Jointly localizing and describing events for dense video captioning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7492–7500, 2018.

[34] R. Krishna, K. Hata, F. Ren, L. Fei-Fei, and J. Carlos Niebles, "Dense-captioning events in videos," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 706–715, 2017.

[35] V. Escorcia, F. C. Heilbron, J. C. Niebles, and B. Ghanem, "Daps: Deep action proposals for action understanding," in *European Conference on Computer Vision*, pp. 768–784, Springer, 2016.

[36] Z. Shen, J. Li, Z. Su, M. Li, Y. Chen, Y.-G. Jiang, and X. Xue, "Weakly supervised dense video captioning," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5159–5167, IEEE, 2017.

[37] X. Duan, W. Huang, C. Gan, J. Wang, W. Zhu, and J. Huang, "Weakly supervised dense event captioning in videos," in *Advances in Neural Information Processing Systems*, pp. 3063–3073, 2018.

[38] L. Zhou, Y. Zhou, J. J. Corso, R. Socher, and C. Xiong, "End-to-end dense video captioning with masked transformer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8739–8748, 2018.

[39] J. Wang, W. Jiang, L. Ma, W. Liu, and Y. Xu, "Bidirectional attentive fusion with context gating for dense video captioning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7190–7198, 2018.

[40] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1, p. 334. MIT press Cambridge, 2016.

[41] M.-T. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” *arXiv preprint arXiv:1508.04025*, 2015.

[42] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, *et al.*, “Google’s neural machine translation system: Bridging the gap between human and machine translation,” *arXiv preprint arXiv:1609.08144*, 2016.

[43] M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado, *et al.*, “Google’s multilingual neural machine translation system: Enabling zero-shot translation,” *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 339–351, 2017.

[44] M.-T. Luong, I. Sutskever, Q. V. Le, O. Vinyals, and W. Zaremba, “Addressing the rare word problem in neural machine translation,” *arXiv preprint arXiv:1410.8206*, 2014.

[45] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, “Long-term recurrent convolutional networks for visual recognition and description,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2625–2634, 2015.

[46] J. Gu, J. Cai, G. Wang, and T. Chen, “Stack-captioning: Coarse-to-fine learning for image captioning,” in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[47] P. Pan, Z. Xu, Y. Yang, F. Wu, and Y. Zhuang, “Hierarchical recurrent neural encoder for video representation with application to captioning,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1029–1038, 2016.

[48] H. Yu, J. Wang, Z. Huang, Y. Yang, and W. Xu, “Video paragraph captioning using hierarchical recurrent neural networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4584–4593, 2016.

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

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

[51] S. Banerjee and A. Lavie, “Meteor: An automatic metric for mt evaluation with improved correlation with human judgments,” in *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pp. 65–72, 2005.

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

[53] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo, “Image captioning with semantic attention,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4651–4659, 2016.

[54] C. Wu, Y. Wei, X. Chu, S. Weichen, F. Su, and L. Wang, “Hierarchical attention-based multimodal fusion for video captioning,” *Neurocomputing*, vol. 315, pp. 362–370, 2018.