CMPE 492 - Senior Project II

Final Report

Image Captioning

Supervisor

Venera Adanova

Jury Members

Aslı Gençtav Tansel Dökeroğlu

Team Members

Buğra Aras Çağdaş Cemre Yurtsuz

Ozan Türkmen Abdullah Yağız Özbek

Table of Contents

1. Introduction and Problem Statement

The problem of automatically generating has been an ongoing interest in natural language processing and computer vision research. Called ‘Image Captioning’, this task requires semantic understanding of images and the ability to translate this understanding into words with proper and correct structure. This is a very and it is an arduous task to achieve a nearly working one. Image Captioning generates natural language descriptions according to the content given in an image. It gives a sense of environment to the computers, which combines the elements of Computer Vision and Natural Language Processing. Image captioning has a significant role in various areas such as image indexing, environmental information understanding of human-like AIs like virtual assistants and interpretation of war scenes in a military field. In recent years, deep learning methods have achieved optimum results for caption generation problems. In this report we will discuss related methods and focus on attention mechanisms. Furthermore, we will discuss the advantages and disadvantages of these methods, in the context of industry standard datasets (Flickr8k and COCO) and evaluation criteria (BLEU) in this field.

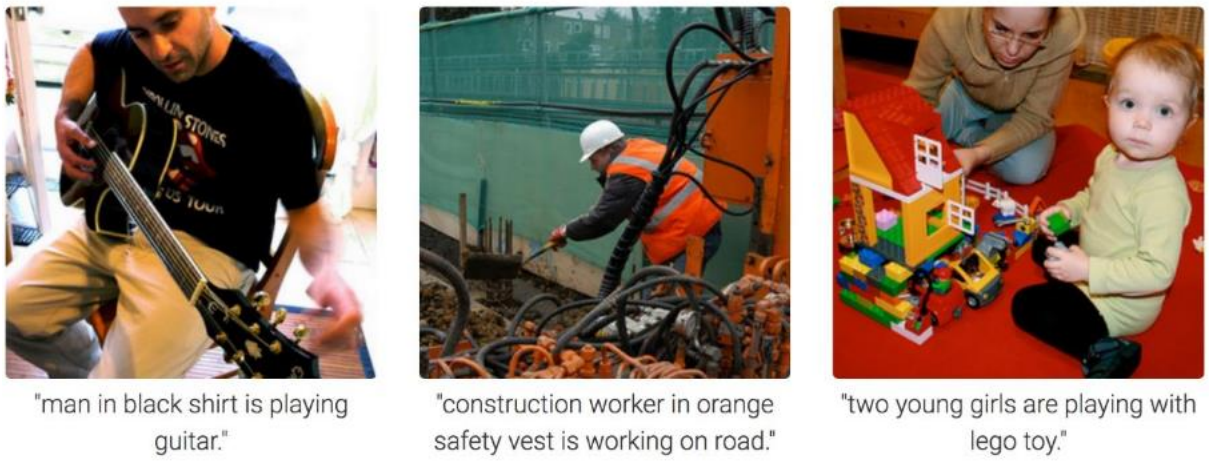


Figure 1. Image Captioning Example [1]

1.1. Background Information

1.1.1. ConvNet/CNN

A Convolutional Neural Network is a Deep Learning algorithm which takes an image as an input, grants importance to the various aspects of the image to differentiate one image to another. Compared to the other classification algorithms, ConvNet requires much lower pre-processing. While in primitive methods filters are hand-engineered, with adequate training, ConvNets can learn these filters and characteristics.

CNNs are composed of multiple layers of artificial neurons. Artificial neurons are mathematical functions that calculate the weighted sum of multiple inputs and output an activation value in a way that they roughly imitate their biological counterparts.

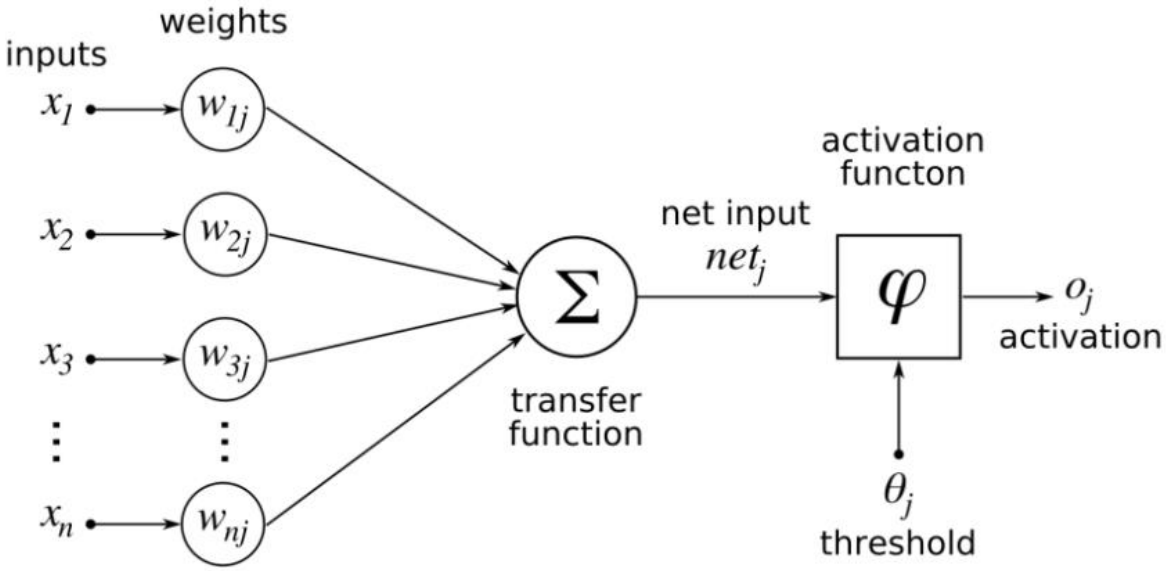


Figure 2. Structure of an artificial neuron [2]

Weights determine the behavior of the neurons. The artificial neuron of a CNN gains various visual features when CNN is fed with the pixel values from images. Activation maps highlight the relevant features of the image. When an image is given to the ConvNet, each of its layers generates several activation maps. Each neuron takes a patch of pixels as input, multiplies their color values by its weights, sums them up, and runs them through the activation function.

The first layer of CNN detects basic features such as diagonal, vertical, and horizontal edges. The output of the first layer is used as input in the next layer, which identifies more complex features such as combination of edges and corners. As we continue to move further in layers of CNN, they detect more complicated features such as faces, objects, animals, flowers, etc.

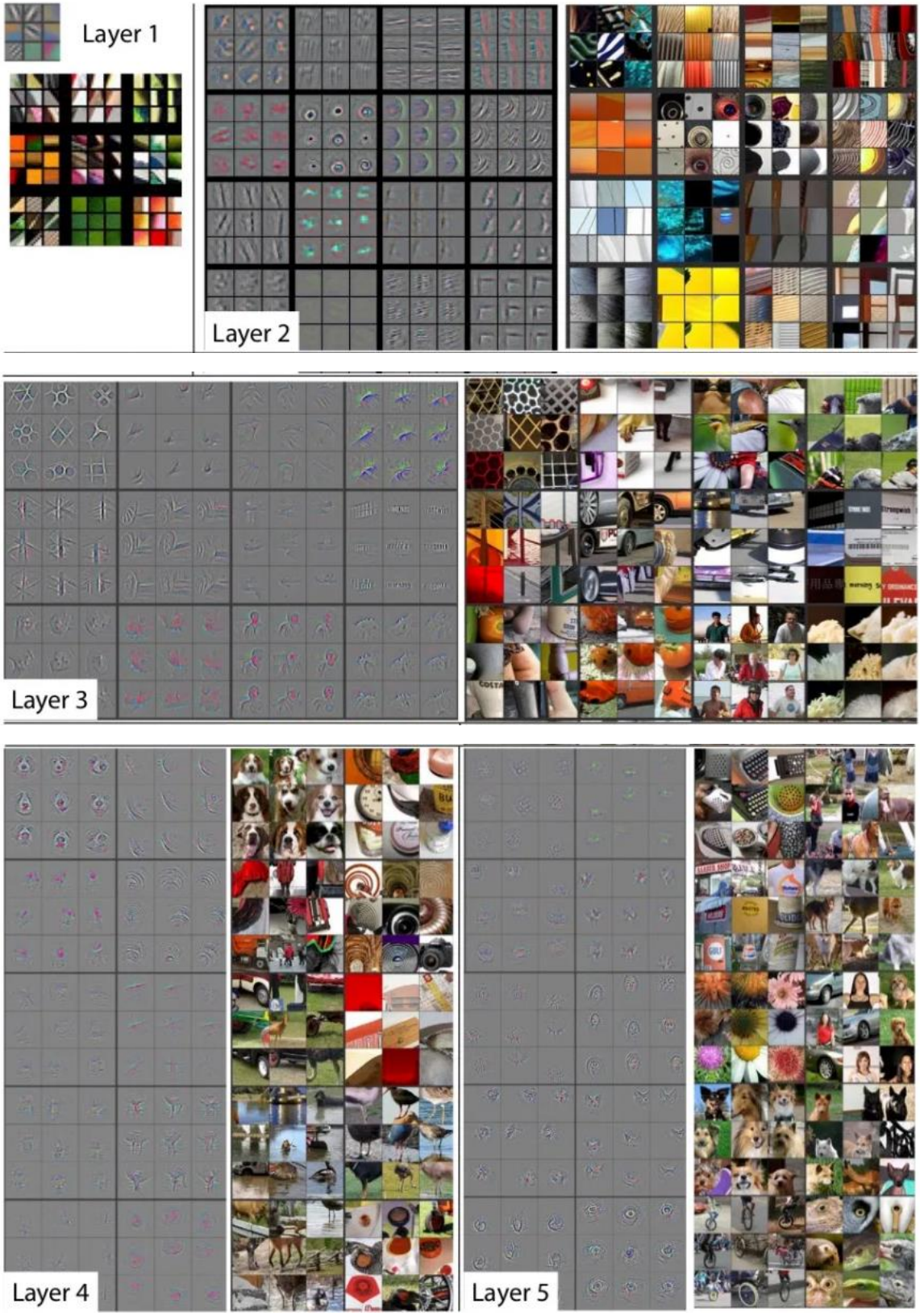


Figure 3. Layers of the neural network [2]

The process of multiplying pixel values by weights and summing them is called "convolution". A CNN is a combination of several convolutional layers but also contains other components such as the final layer. The final layer of a CNN is a classification layer, which takes input as output of the last convolution layer.

The classification layer outputs a set of confidence scores between 0 and 1 based on the activation map of the last convolution layer. Scores define how likely the image belongs to a class. Final layer basically computes the set of possibilities of classes and the softmax layer outputs the highest possibility of a specific class as an output.

When it comes to training CNNs, we did not take part in its implementation since there are many CNN models that are proven to give best results and training CNNs takes so much time and resources. In this project we used ResNet152V2 and XCeption CNN models.

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88

Figure 4. Some Keras Applications [3]

In our project we do not need the softmax layer in our CNN model since we need all the possible classification possibilities to generate a caption. Imagine an image that has a car on the road, inside the car there is a dog, and, in the background, there is a forest. A generic CNN model might classify this image as a dog, car, or forest. However, we need all the elements there are in the image to create a good sentence that consists of words dog, car, road, and forest. To achieve that, we just removed the last layer of CNN.

1.1.2. RNN (LSTM)

Recurrent Neural Networks are feedforward neural networks that also have an internal memory. RNN performs the same function for every input data and output of the current input depends on the previous computations. After output is generated, it is copied and sent back to the recurrent network. To make a prediction, it considers current input and previous output that is learnt from previous input.

What differs RNNs from feedforward neural networks is that they can use their internal memory to process sequence of inputs. This feature of RNNs helps us to generate captions. In RNN, all inputs are dependent on each other but in other neural networks they are not.

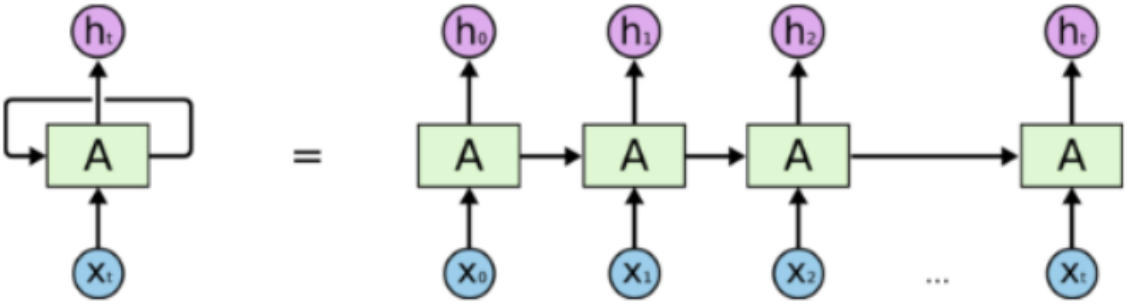


Figure 5. An unrolled recurrent neural network [4]

Basically, formula for the current state is:

$$h_t = f(h_{t-1}, x_t)$$

Applying Activation Function:

$$h_t = \tanh (W_{hh}h_{t-1} + W_{xh}x_t)$$

W is weight, h is the hidden vector, W_{hh} is the weight at previous hidden state, W_{xh} is the weight at current input state, tanh is the Activation function, that implements a non-linearity that squashes the activations to the range [-1,1].

Output:

$$y_t = W_{hy}h_t$$

Where y_t is the output state and W_{hy} is the weight at the output state.

However, gradient vanishing is a huge disadvantage in RNN. Fortunately, LSTM solves this problem. Long Short-Term Memory networks are an altered version of RNNs, which are better at remembering and keeping larger memory. LSTM trains model by back-propagation.

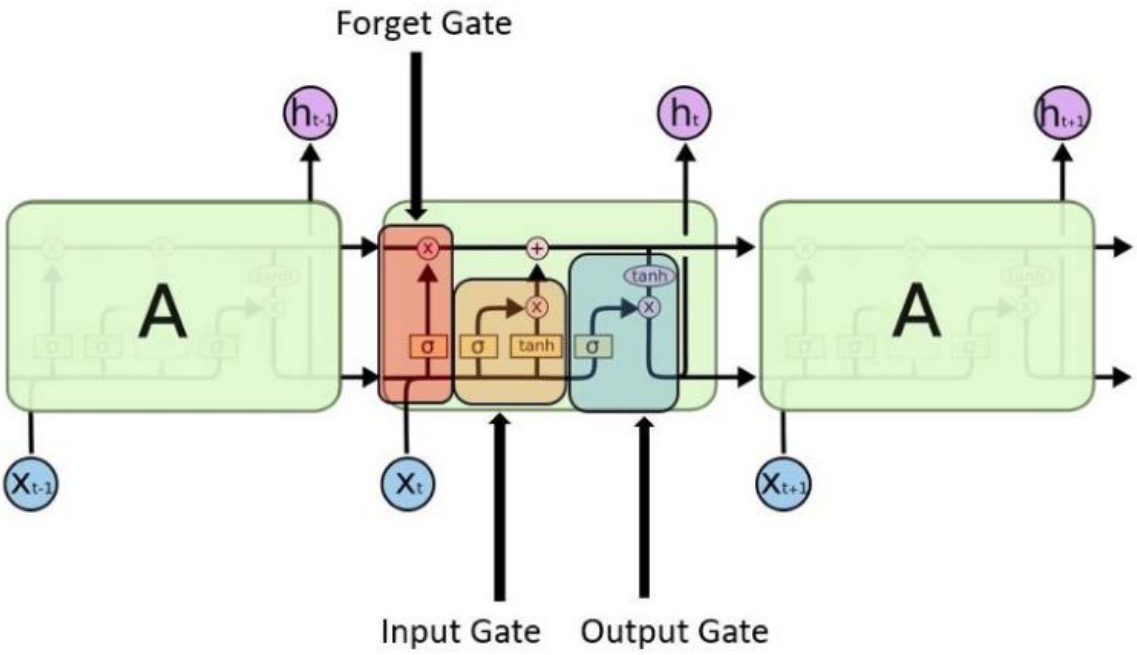


Figure 6. An unrolled recurrent neural network [4]

A generic LSTM network consists of different memory blocks called A cells. There are 2 outputs that are being transferred to the next cell. The cell state and hidden state (h_n). Inside the cells, there are 3 gates that are responsible for memory operations. These gates are:

- Forget Gate

Forget gate is responsible for removing data from the memory of the LSTM. The data that is no longer important for the context of the sentence is removed via multiplication of a filter. This process is required to improve the performance of the LSTM.

- Input Gate:

The input gate adds new data to the cell state. This addition process is simply a 3-step process.

1. Filtering what values will be added to the cell state by using sigmoid function. This step is very similar to the forget state.
2. Generating a vector that contains all possible values that can be added to the cell state. This process is done by the tanh function, which outputs values [-1,1].
3. Multiplying the value of the regulatory filter to the generated vector and then adding this information to the cell state by addition.

- Output Gate:

The output gate selects information from the current cell state and outputs it. The processes of output gate can be explained in 3 states:

1. After applying tanh function to the cell state, it generates a vector. That way scales the values of range -1 to 1.
2. To regulate the values that need to be output from the vector generated above, it creates a filter by using the values h_{t-1} and X_t.
3. By multiplying the vector created in step 1 and value of this regulatory filter and sending it out as an output and to the hidden state of the next cell.

1.1.3. Embedding

An embedding is a mapping of a discrete vector of continuous numbers. In the neural networks field, embeddings are low-dimensional, learned continuous vector representations of discrete variables. In the context of English words, embeddings help us to give meaning to these words to create meaningful sentences. However, not all embedding algorithms give information about the relationship between words (such as synonyms).

Glove is a model for distributed word presentation. It is an unsupervised learning algorithm that achieves vector representations for words. In Glove, words mapped into a 2D space where distance between words is related to semantic similarity.

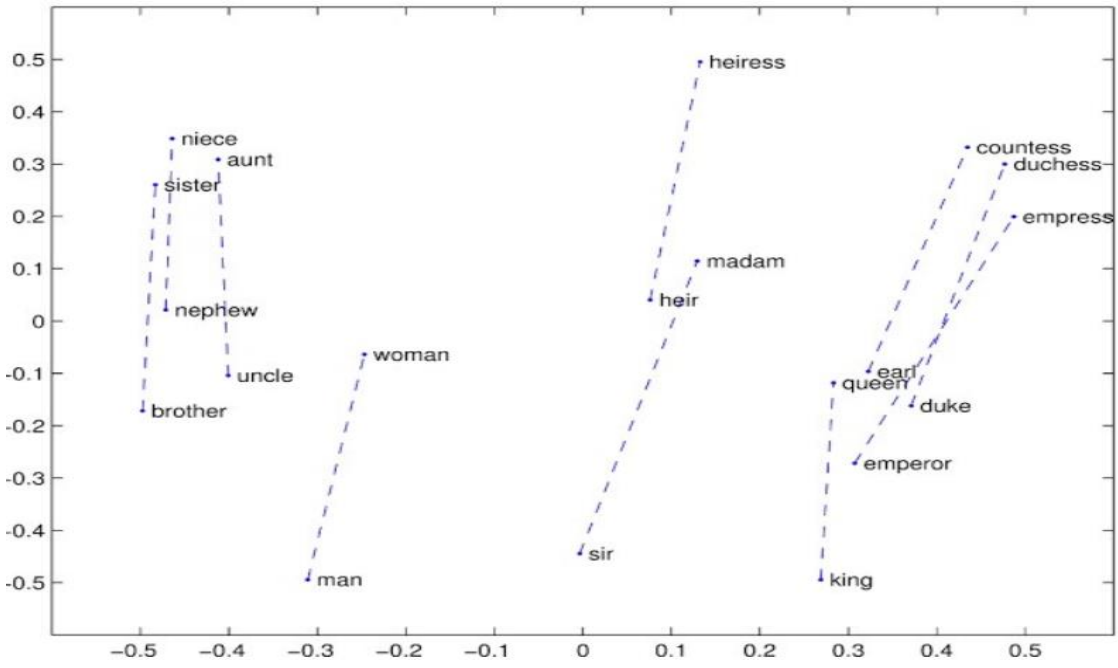


Figure 7. Embedding Space [5]

We use Glove and LSTM together because we cannot give words as String to the LSTM as input. Glove turns these words into a one hot vector which LSTM can interpret. Relationship between them looks like this:

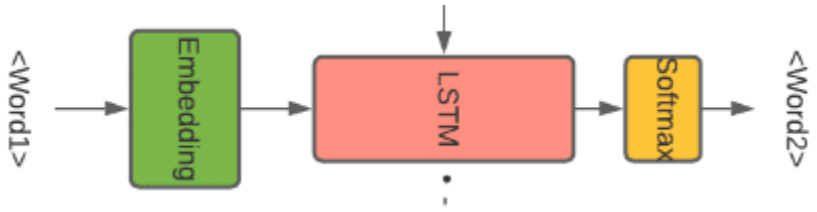


Figure 8. Embedding-LSTM relation

1.1.4. Attention

In the field of neural networks, attention is a technique that mimics cognitive attention. It increases the value of important parts of the input and reduces the rest. It decides which parts are important and are not by learning through gradient descent. Attention mechanisms are mainly created for Language Transition. Later, people started to use it in Computer Vision and Image Captioning and so on. There are 2 types of Attention models that we are interested in:

- **Global Attention**

To compute the output, input from every encoder state and decoder states prior to the current state is used.

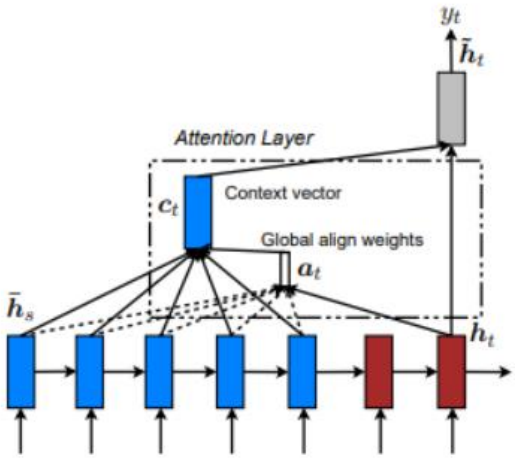


Figure 9. Global Attention [6]

a_t attention weights are calculated using each encoder step and h_t decoder previous step. Then taking the product of Global align weights and each encoder steps with the use of a_t, the context vector is created. Finally, it is fed to the LSTM cell to find decoder output.

Below is the mathematical presentation of what is explained above in the “general” part of the formula:

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W}_{\boldsymbol{a}} \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W}_{\boldsymbol{a}}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & \textit{concat} \end{cases}$$

Figure 10. Formula [7]

- **Local Attention**

Difference between Global and Local attention is that local attention uses only a few positions from the encoder to calculate a_t.

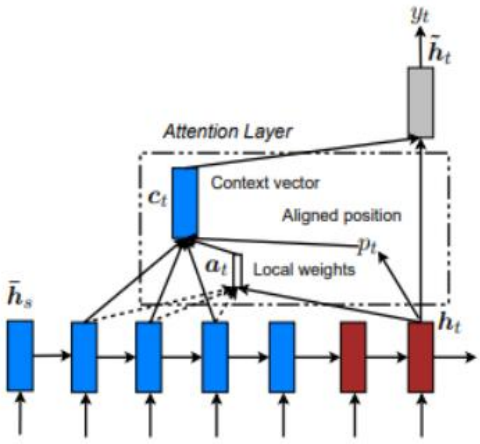


Figure 11. Local Attention [6]

First single-aligned position p_t is found then a window of words from the encoder layer along with h_t is used to calculate align weights and context vector. Then, as global attention, the context vector is fed to the LSTM.

$$e_{ij} = v_a^\top \tanh(W_a s_{i-1} + U_a h_j)$$

$$a_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})}$$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

Figure 12. Local Attention Formula [7]

Here is what happens in local attention. h_j is jth hidden state we derive from the encoder, s_{i-1} is the previous time step's hidden state, and W,U and V are all weight matrices that are learnt while training. Then, we normalize values through softmax which gives values between 0 and 1. Thus, this scoring function provides probabilities that determine how important each hidden state is for the current time step. Finally, each encoder's hidden state and corresponding score is multiplied and then summed all to create the context vector.

In this project we used Local Attention mechanism to reduce computation time.

2. Development History

We used different neural network architectures throughout our project and tried many different methods to increase the performance of these architectures. The first architecture we followed for the model includes a pretrained CNN and an RNN that will work as encoder, and there is a feedforward network that will work as a decoder and process the merged vectors of two different networks in the encoder.

In this first architecture we implemented, we learned from our research that we would get better results by changing the way we select different predictions from the model and create hypothesis captions. These selection methods are Greedy Search and Beam Search.

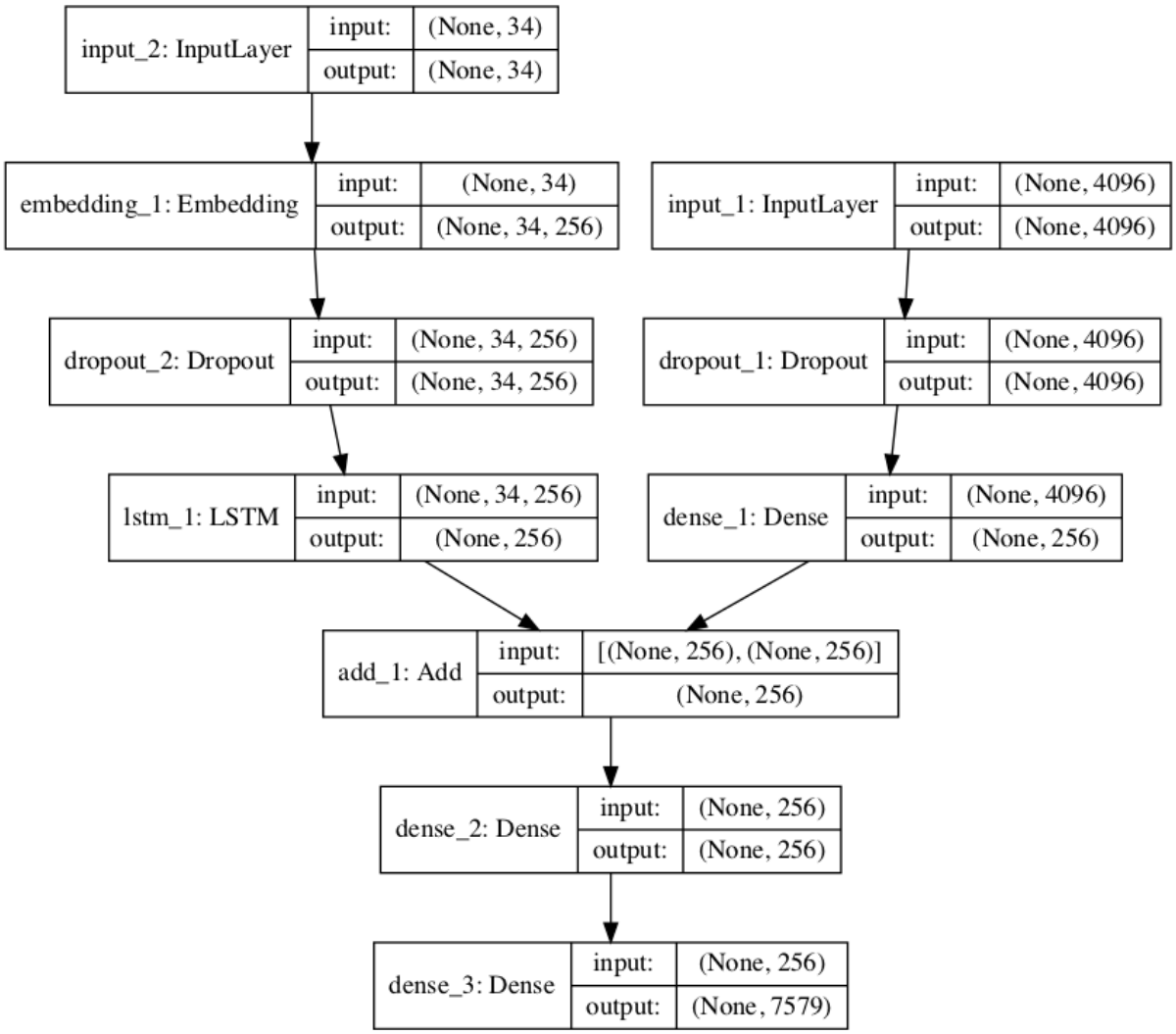


Figure 13. Earlier System [8]

2.1. Greedy Search Approach

The algorithm makes the optimal choice at each step as it attempts to find the overall optimal way to solve the entire problem. In our case the problem the Greedy Search approach is trying to solve is getting the most probable word from the model. So basically, in this approach we just add the most probable word the model provides us to the predicted caption.

2.2. Beam Search Approach

"The local beam search algorithm keeps track of k states rather than just one. It begins with k randomly generated states. At each step, all the successors of all k states are generated. If anyone is a goal, the algorithm halts. Otherwise, it selects the k best successors from the complete list and repeats." [7]

Instead of greedily choosing the most likely word as the sequence is constructed, the beam search keeps the *k* most likely words, where *k* is a user-specified parameter and controls the number of beams or parallel searches through the sequence of probabilities. The search process can halt for each candidate separately either by reaching a maximum length, by reaching an end-of-sequence token, or by reaching a threshold likelihood.

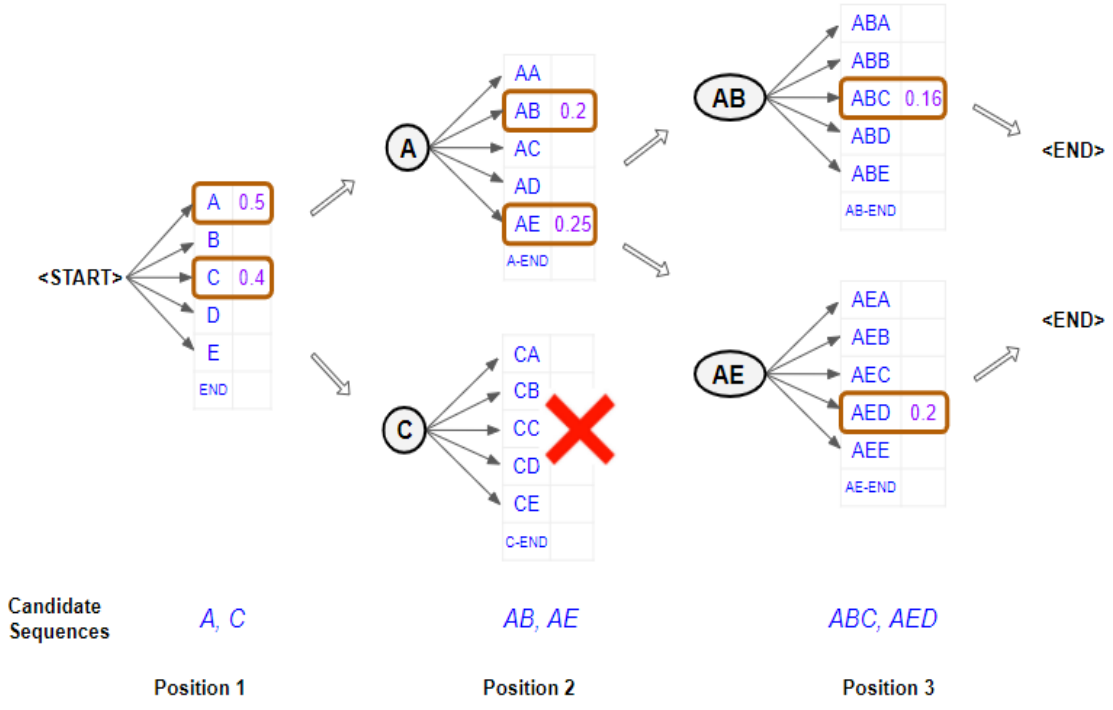


Figure 14. Beam Search [9]

2.3. Issues of The Earlier System

The program we developed has many lackingllows:

- Inadequate data size causes generic captions.
- Small vocabulary, which is ns (man, woman, two, etc.) causes those words to become outliers and it drops the accuracy of the model. For example, the caption for the image that contains only a car, road and trees is “A man with blue shirt walks in forest.”.

2.4. Attention Mechanism

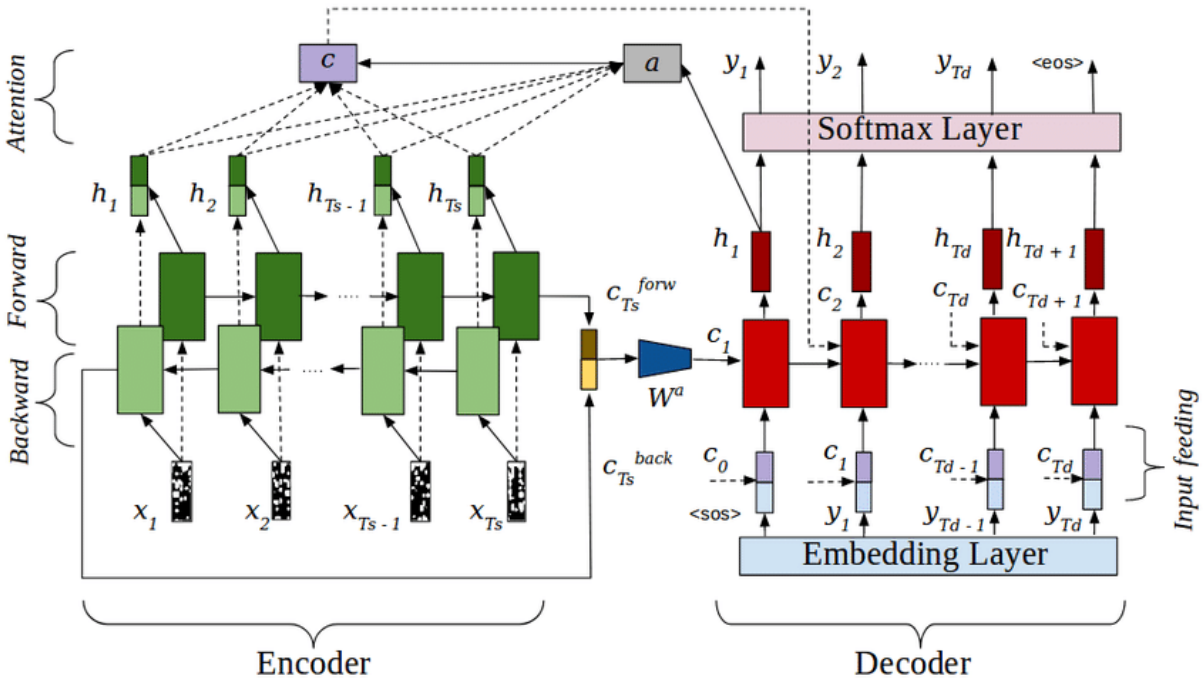


Figure 15. Attention Mechanism [10]

As we found in our studies to improve the performance problems we encountered in the previous system, we decided to try attention mechanism, which is one of the most popular applications especially in the field of image captioning. a points due to lack of data size and hardware insufficiency. Pro Caused by lack of data size. Small vocabulary size hinders the diversity of generated captions.

Overuse of some words in descriptions caused problems with generalization and made it harder for the model to relate between a word and different objects in an image. The attention mechanism enabled the features of multiple objects in the image to be mapped better with words, the context, and details to be better understood by the model, hence, more specific captions to be

created. Also, since attention makes use of only part of the input, rather than the whole static image, when making predictions, the amount of time spent to train the model greatly decreased.

3. Final Architecture

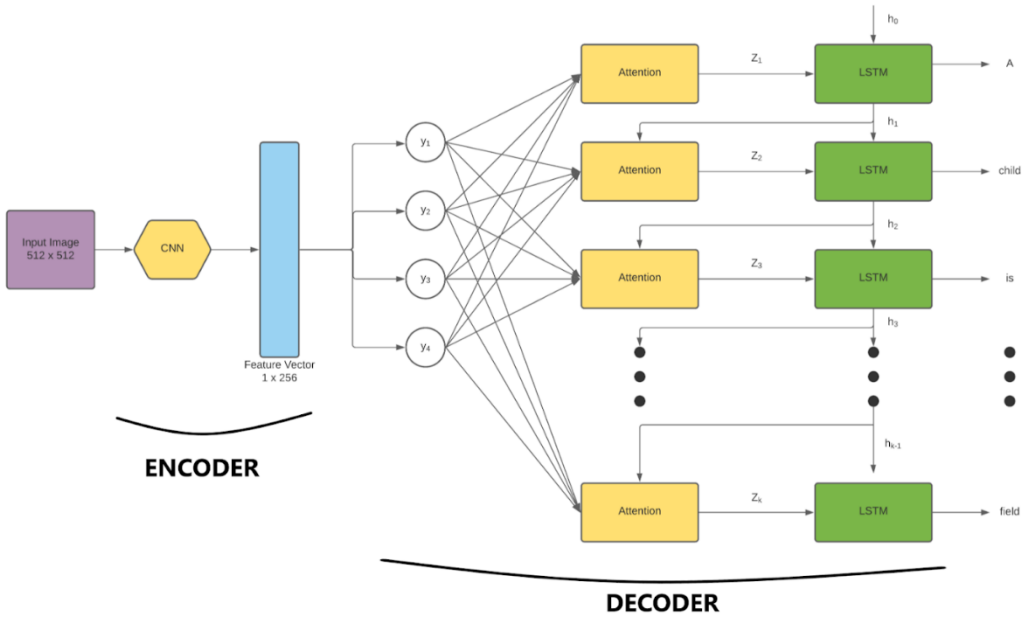
The final state of our project consists of 2 parts which are encoder and decoder. Our system first encodes the image by using pre-trained CNN model Resnet152V2 which outputs a feature vector that is called hidden state in encoder-decoder systems. Then, the decoder generates a caption by using this hidden state by using local attention and LSTM.

To generate the next sequence in the caption, previous outputs and new sequence data are used as input. We mentioned in the background information section in our report that it gives RNN networks an internal memory which we believe helps to generate captions that are more informative and context aware.

However, RNN networks take too much memory and time to train and evaluate. This causes us to limit the memory of our LSTM. Attention models solve this problem by filtering most relevant information from the input image.

The image is first divided into n parts with the attention mechanism, and we compute an image representation of each word. When the RNN is generating a new word, the attention mechanism focuses on the relevant part of the image, so the decoder uses only this part of the image.

The context vector is generated with the use of the hidden states of both the encoder and the decoder. Then the attention mechanism aligns the input and output sequences, with an alignment score parameterized by a feed-forward network. This helps to pay attention to the most relevant information in the sequence. After that, the model predicts a target word (see. greedy search approach) based on the context vectors associated with the source position and the previously generated target words.



4. Tests and Results

As a result of our early research, we learned that the most common and effective method used for caption scoring in the field of Image Captioning, including our project, is to use the BLEU (bilingual evaluation understudy) algorithm. In this period, we made our performance measurements by taking 1,2,3,4-gram BLEU scores of the captions extracted by our model.

We have also considered the BLEU scores of other individuals and institutions that have worked in this field while conducting our tests and developments. In this way, we were able to notice possible implementation errors.

In addition to the BLEU algorithm, we gave random photos from the validation sets as input to the model and examined the outputs manually. Manual examinations were made by comparing the image descriptions of other people in the data set, our descriptions, and the image we gave as input with the descriptions we received from the model.

We made measurements such as the quality of our captions and the training speed of the model with different model types and dataset combinations. After each measurement, we chose the model structure and data set that gave the best measurement and continued our development with these choices.

The models we use and the table we created with two datasets with the best results are as follows:

4.1. Different Systems using Flickr8k

	Seq2Seq (Greedy)	**Seq2Seq (Beam Search)**	**Attention**	**Attention + Glove**
BLEU-1	0.5422641509433962	0.5183691363437088	0.6415317804974339	0.6734793187347932
BLEU-2	0.35531095204272756	0.3355465247440823	0.5586815031689765	0.5994045265309978
BLEU-3	0.23107431773554588	0.22012286802645842	0.4883667559632731	0.5338080623317552
BLEU-4	0.14555085440385526	0.13933965485676503	0.41846500210706394	0.4701072022898008
Training Time (Seconds)	2372		1142	

4.2. Different Datasets with Attention and Glove System

	Flickr8k	**COCO**
BLEU-1	0.6734793187347932	0.4002164990384127
BLEU-2	0.5994045265309978	0.2282096837528966
BLEU-3	0.5338080623317552	0.13193782648046362
BLEU-4	0.4701072022898008	0.0713707516535882
Training Time (Seconds)	1142	5400

As a result of our tests, we saw that the Encoder Decoder model with Attention mechanism gave the best results when trained with Flikr8k dataset.

5. Resources

Initially and throughout the lifecycle of our project, we were constantly on a state of research for tutorials, documentations, similar designs, and implementations.

5.1. Tutorials, Papers and Documentations

- Keras RNN Guides [11]
- TensorFlow Image Captioning Tutorials [12]
- Oscar: Object-Semantics Aligned Pre-training for Vision-Language Tasks [13]
- Visual Commonsense R-CNN [14]
- Reflective Decoding Network for Image Captioning [15]
- VirTex: Learning Visual Representations from Text Annotations [16]
- TensorFlow API Python Docs [17]
- Keras API Docs [18]

5.2. Similar Designs

These are the designs that we analyzed carefully and got inspired from.

- **Image Captioning: Image to Text [19]**
  by Ashutosh Soni

- **Image Captioning with Keras [20]**
  by Harshall Lamba

- **Image Captioning using Attention Mechanism [21]**
  by Subham Sarkar

- **Multi-Modal Methods: Image Captioning (From Translation to Attention) [22]**
  by MTank

6. Impacts

In the past few years, the problem of generating descriptive sentences for images has increased interest in Deep Learning. Deep learning is a subcategory of machine learning. With both deep learning and machine learning, algorithms are learning. This is done with the algorithms that analyze big amounts of data and then take actions or perform a function based on gathered information from that data. Our project, Image Captioning, is a huge area in Deep Learning. It has been a very important and fundamental task in the Deep Learning area.

Image captioning and Deep learning is used in many areas such as, health, economy, military. We can see its applications in our daily life. Many social media applications use image captioning for identifying images. Facebook, Google, and Apple recognize objects in photos and categories them in albums. Tesla and other automobile brands use this technology for recognizing traffic signs, pedestrians, and other cars. Image captioning and deep learning have a huge impact on the medicine area. There are applications that can detect cancerous cells in the body. NVIDIA is using image captioning technologies to create an application to help people who have low or no eyesight.

On the other hand, there are some concerns about the Ethics of image captioning and deep learning. Face recognition or image captioning describe objects and people with their color so, sometimes these technologies can generate racist captions. Also with these applications, our personal data could be cached in databases to process and train the data, so when our data is publicly stored in databases for the training of new models, it may pose a risk of our personal data being stolen.

7. Future Work and Conclusion

At the end, with our system, we have accomplished results close to the state-of-the-art. We have learnt a great deal regarding machine learning and natural language processing. Some of the future work that we can think of right now would be:

- Doing more hyper parameter tuning on learning rate, batch size, number of layers, number of units, dropout rate and batch normalization
- Using a higher end system to improve preparation speed
- Make an API version using something like Flask and deploy it in a cloud.

8. References

[1] https://towardsdatascience.com/image-captioning-in-deep-learning-9cd23fb4d8d2

[2] https://bdtechtalks.com/2020/01/06/convolutional-neural-networks-cnn-convnets

[3] https://keras.io/api/applications

[4] https://aditi-mittal.medium.com/understanding-rnn-and-lstm-f7cdf6dfc14e

[5] https://medium.com/analytics-vidhya/word-vectorization-using-glove-76919685ee0b

[6] https://arxiv.org/pdf/1508.04025.pdf

[6] https://machinelearningmastery.com/develop-a-deep-learning-caption-generation-model-in-python

[7] https://ai.plainenglish.io/introduction-to-attention-mechanism-bahdanau-and-luong-attention-e2efd6ce22da

[8] Pages 125-126, Artificial Intelligence: A Modern Approach (3rd Edition), 2009.

[9] https://towardsdatascience.com/foundations-of-nlp-explained-visually-beam-search-how-it-works-1586b9849a24

[10] https://www.researchgate.net/publication/326570264_An_Efficient_End-to-End_Neural_Model_for_Handwritten_Text_Recognition

[11] https://keras.io/guides/working_with_rnns

[12] https://www.tensorflow.org/tutorials/text/image_captioning

[13] https://paperswithcode.com/paper/oscar-object-semantics-aligned-pre-training

[14] https://paperswithcode.com/paper/visual-commonsense-r-cnn

[15] https://paperswithcode.com/paper/reflective-decoding-network-for-image

[16] https://paperswithcode.com/paper/virtex-learning-visual-representations-from

[17] https://www.tensorflow.org/api_docs/python

[18] https://keras.io/api

[19] https://medium.com/image-recreation-a-method-to-make-learning-of-gan/image-captioning-image-to-text-ba5fb5754625

[20] https://towardsdatascience.com/image-captioning-with-keras-teaching-computers-to-describe-pictures-c88a46a311b8

[21] https://medium.com/swlh/image-captioning-using-attention-mechanism-f3d7fc96eb0e

[22] https://blog.mlreview.com/multi-modal-methods-image-captioning-from-translation-to-attention-895b6444256e