# Image captioning Midterm Presentation

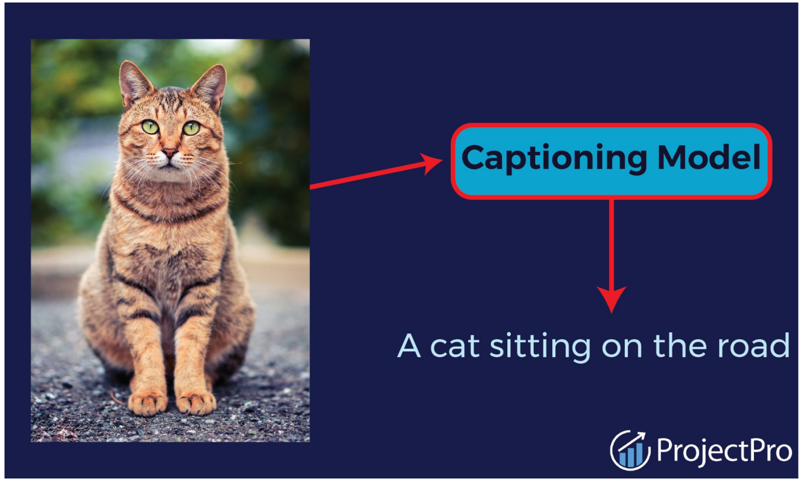
Group 6

## Outline

## 1. Introduction to Image Captioning

Caption generation is a challenging artificial intelligence problem where a textual description must be generated for a given photograph or a picture.

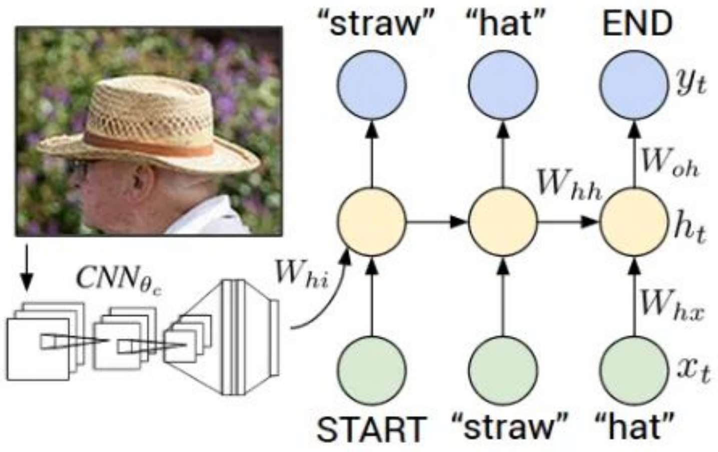
It requires both methods from computer vision to understand the content of the image and a language model from the field of natural language processing to turn the understanding of the image into words in the right order.



## Architecture

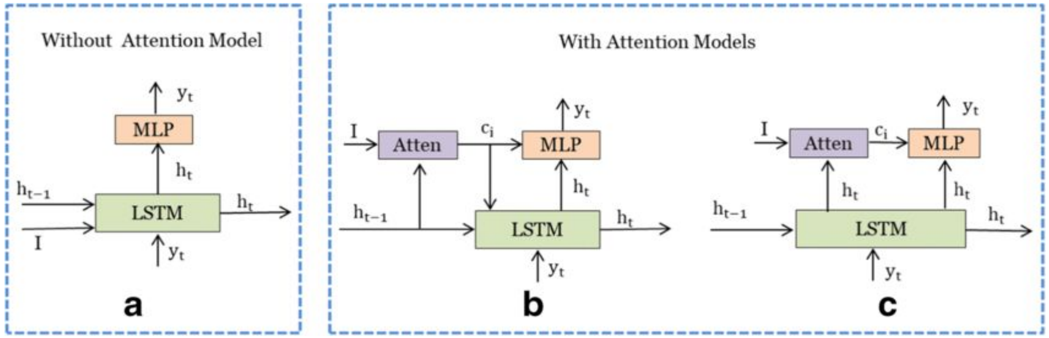
A typical image captioning model consists of encoder and decoder. The Convolutional Neural Network(CNN) is the encoder in which input images is given to CNN to extract image features. The last hidden state of CNN is connected to Decoder.

The decoder is the Recurrent Neural Network(RNN) which does language modelling up to the word level. The first time step receives the encoded output from the encoder and also the <START> vector.



## 2. Problem with classical Encoder-Decoder architecture

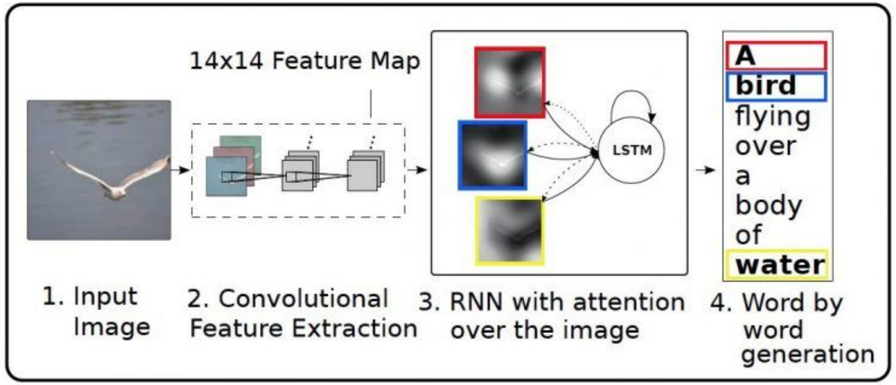
The problem with this method is that, when the model is trying to generate the next word of the caption, this word is usually describing only a part of the image. It is unable to capture the essence of the entire input image. Using the whole representation of the image h to condition the generation of each word cannot efficiently produce different words for different parts of the image.This is exactly where an Attention mechanism is helpful.



## Attention mechanism

With an Attention mechanism, the image is first divided into *n* parts, and we compute with a Convolutional Neural Network (CNN) representations of each part *h1,…, hn.*

When the RNN is generating a new word, the attention mechanism is focusing on the relevant part of the image, so the decoder only uses specific parts of the image.


14x14 Feature Map
LSTM
A
bird
flying
over
a
body
of
water
1. Input Image
2. Convolutional Feature Extraction
3. RNN with attention over the image
4. Word by word generation

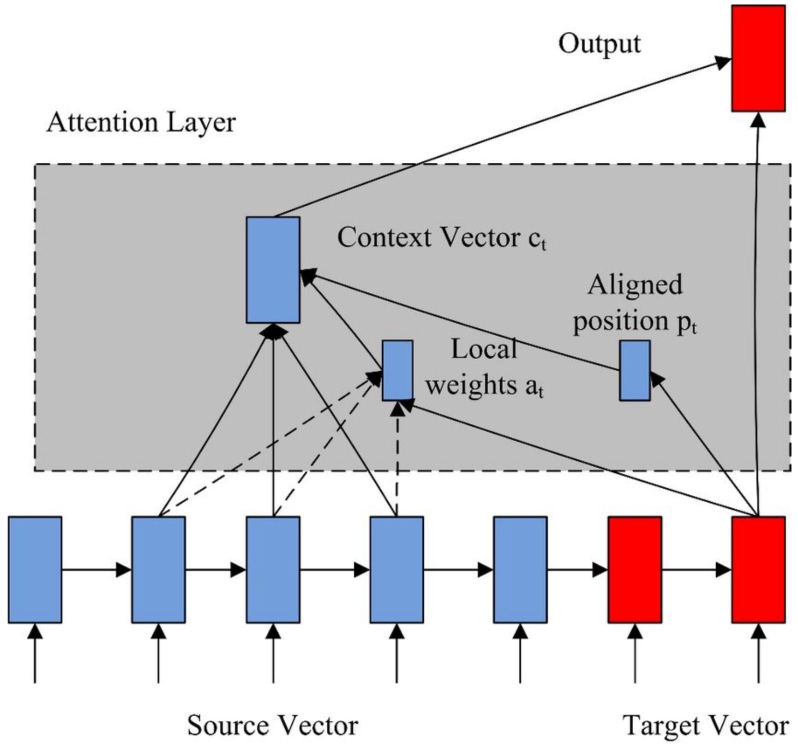
## More about Attention mechanism

If we have predicted *i* words, the hidden state of the LSTM is *hi*. We select the « relevant » part of the image by using *hi* as the context. Then, the output of the attention model *zi*, which is the representation of the image filtered such that only the relevant parts of the image remains, is used as an input for the LSTM. Then, the LSTM predicts a new word and returns a new hidden state *hi+1*.

## Local Attention (BAHDANAU)

In our model, we will use Local Attention. As Global attention focus on all source side words for all target words, it is computationally very expensive and is impractical when translating for long sentences.

To overcome this deficiency local attention chooses to focus only on a small subset of the hidden states of the encoder per target word.



## Attention mechanism mathematical formula

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

*where ,*

1. $e_{jt}$ means at every $t^{th}$ timestep of **DECODER,** how important $j^{th}$ is the pixel location in the input image.
2. $s_{t-1}$ is previous state of **DECODER**.
3. $h_j$ is the state of **ENCODER**.
4. $f_{ATT}$ is a simple Feed Forward Neural Network, which is a linear transformation of input ($U_{attn} * h_j + W_{attn} * s_t$) and then a non-linearity(tanh) on top of that, and then again one more transformation ($V^T_{attn}$). It is a **SCALAR** quantity

## Attention mechanism mathematical formula

Local attention formula

$$f_{ATT} = V_{attn}^{T} * tanh(U_{attn} * h_j + W_{attn} * s_t)$$

where,

$$V_{attn}^{T} \in R^{d}\ , U_{attn} \in R^{d*d}\ , W_{attn} \in R^{d*d}, s_{t-1} \in R^{d}\ , h_j \in R^{d}$$

## Attention mechanism mathematical formula

To get the Probability distribution we do Softmax :

$$\alpha_{jt} = Softmax(e_{jt})$$

i.e. $$\alpha_{jt} = \frac{e^{e_{jt}}}{\sum_{k=1}^{T_x} e^{e_{kt}}},\ \text{such that}\ \sum_{j=1}^{T_x} \alpha_{jt} = 1\ \text{and},\ \alpha_{ij} \geq 0$$

Now, when we know the input , we need to feed Weighted Sum combination of input to **DECODER**.

## Attention mechanism mathematical formula

$$C_t = \sum_{j=1}^{T} \alpha_{jt} h_j \; \textit{such that} \; \sum_{j=1}^{T_x} \alpha_{jt} = 1 \; \textbf{\textit{and,}} \; \alpha_{ij} \geq 0$$

*where ,*

$C_t$ is the **context vector**, i.e. the Weighted Sum of input.

## Attention mechanism mathematical formula

$$s_t = RNN(s_{t-1}, [e(\hat{y}_{t-1}), c_t])$$

*where ,*

$s_{t-1}$ is previous state of **DECODER.**

$e(\hat{y}_{t-1})$ is previous predicted word.

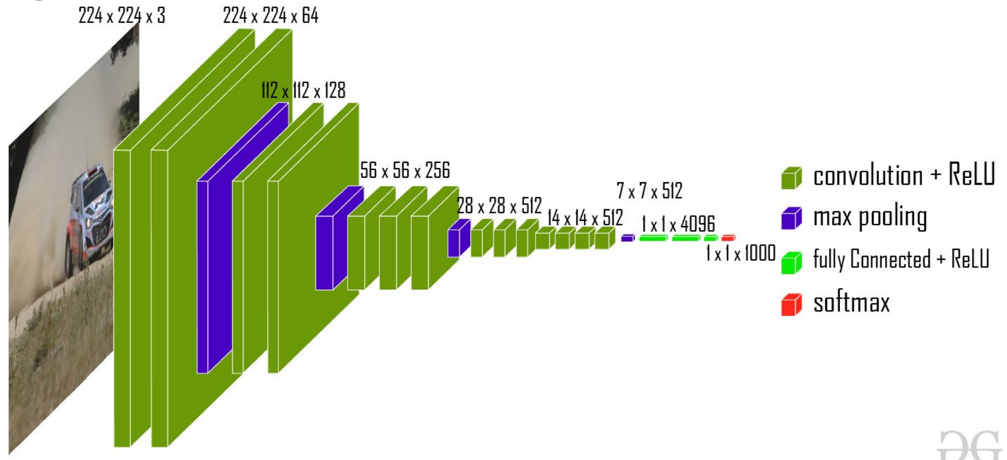
$c_t$ is the **context vector**, i.e. the Weighted Sum of input

## Encoder model - VGG16

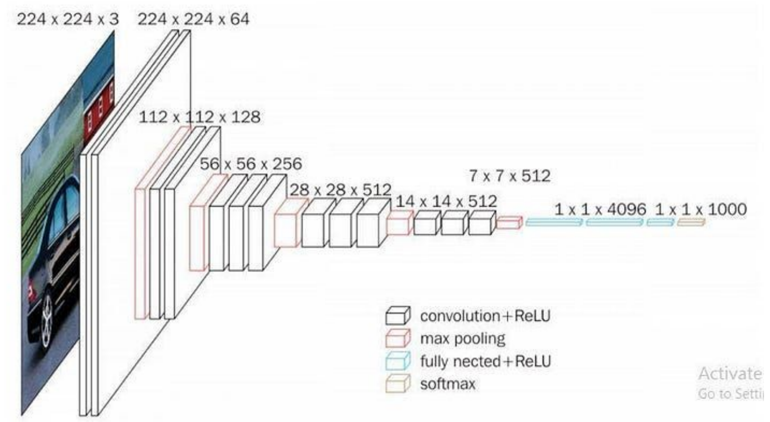
- The 16 in VGG16 refers to 16 layers that have weights. In VGG16 there are thirteen convolutional layers, five Max Pooling layers, and three Dense layers which sum up to 21 layers but it has only sixteen weight layers i.e., learnable parameters layer.
- VGG16 takes input tensor size as 224, 244 with 3 RGB channel
- Most unique thing about VGG16 is that instead of having a large number of hyper-parameters they focused on having convolution layers of 3x3 filter with stride 1 and always used the same padding and maxpool layer of 2x2 filter of stride 2.
- The convolution and max pool layers are consistently arranged throughout the whole architecture
- Conv-1 Layer has 64 number of filters, Conv-2 has 128 filters, Conv-3 has 256 filters, Conv 4 and Conv 5 has 512 filters.
- Three Fully-Connected (FC) layers follow a stack of convolutional layers: the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer.

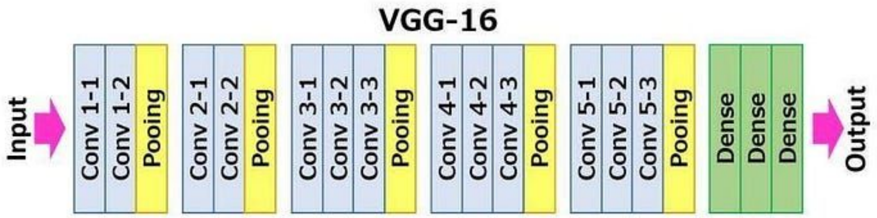
## Encoder model - VGG16

VGG 16 was proposed by Karen Simonyan and Andrew Zisserman of the Visual Geometry Group Lab of Oxford University in 2014 in the paper “VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION”. This model won 1st  and 2nd place in the above categories in the 2014 ILSVRC challenge



## Encoder model - VGG16

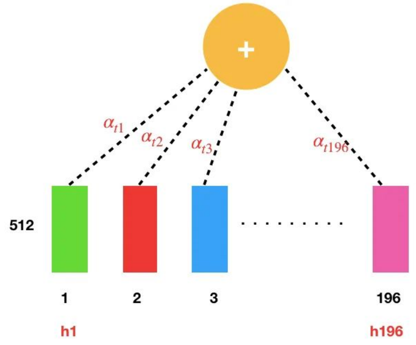




## Encoder model - VGG16

The output of the 5th convolution layer of VGGNet is a **14*14*512** size feature map.This 5th convolution layer has 14*14 pixel locations which corresponds to certain portion in image, that means we have 196 such pixel locations.

The model will then learn an attention over these locations(**which in turn corresponds to actual locations in the images**). As shown in the below figure 5th convolution block is represented by 196 locations which can be passed in different time step.



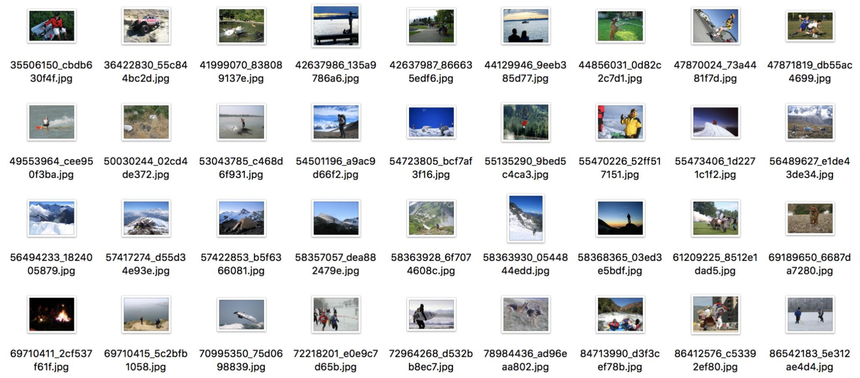
## Decoder model - LSTM

LSTM stands for Long Short-Term Memory, which is a type of recurrent neural network (RNN) used in deep learning. LSTM networks are designed to handle the vanishing gradient problem that regular RNNs face by using a memory cell to store information over a longer period of time. The memory cell is responsible for maintaining long-term dependencies in the input sequence, while gating mechanisms within the network layer enable the model to selectively update or forget information. This allows LSTM networks to effectively learn and predict from sequences of data with complex temporal dependencies, making them commonly used in applications such as speech recognition, language modeling, and video processing.

## 3. Dataset

For training, we will use the Flickr8k dataset. It is a labeled dataset consisting of 8000 photos with 5 captions for each photos. It includes images obtained from the Flickr website.

A new benchmark collection for sentence-based image description and search, consisting of 8,000 images that are each paired with five different captions which provide clear descriptions of the salient entities and events.



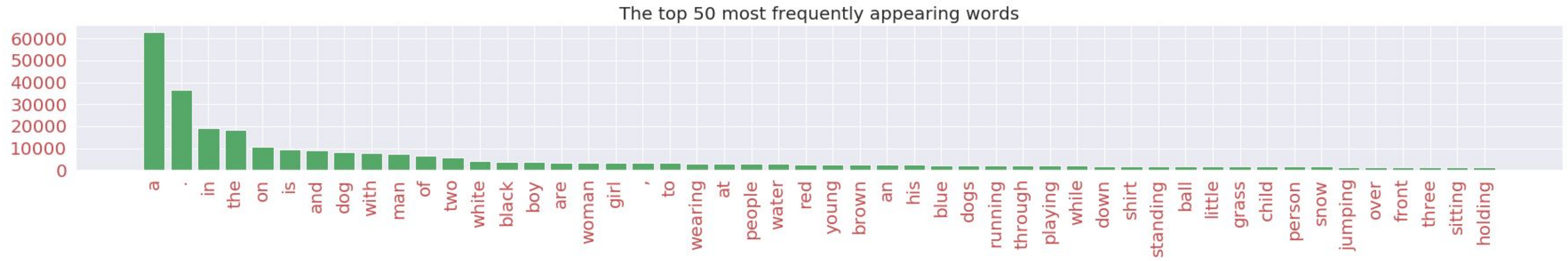
## Preprocess

Preprocess caption:

In this step, the main objective is to load captions and preprocess text. It will remove any characters that are not in plain alphabetic character, uppercase letters are lowered.

Then the frequency of words are found and ordered by how much the words appear.

Frequent words such as “a” or “the” don’t hold much importance and don’t describe much context, therefore these words are removed.



## Model used

CNN ENCODER:

VGG16 is used to make the CNN encoder. The encoder consists of a single fully connected layer followed by ReLU activation function.

## Model used

**RNN DECODER and BAHDANAU ATTENTION**

Attention is used in the RNN Decoder. This Attention Mechanism will return the context vector and attention weights over the time axis.

## Step by step of local decoder using Attention

1. **Producing the Encoder Hidden States** — Encoder produces hidden states of **each** element in the input sequence
2. **Calculating Alignment Scores** between the previous decoder hidden state and each of the encoder's hidden states are calculated *(Note: The last encoder hidden state can be used as the first hidden state in the decoder)*
3. **Softmaxing the Alignment Scores** — the alignment scores for each encoder hidden state are combined and represented in a single **vector** and subsequently **softmaxed** to produce attention weight.
4. **Calculating the Context Vector** — the encoder hidden states and their respective alignment scores are *multiplied* to form the **context vector**
5. **Decoding the Output** — the context vector is *concatenated* with the previous decoder output and fed into the **Decoder RNN** for that time step along with the previous decoder hidden state to produce a **new output**
6. The process (steps 2–5) **repeats** itself for each time step of the decoder until an token is produced or output is past the specified maximum length

## Training

**Training Step:**

- The ENCODER output, hidden state(initialised to 0) and the DECODER input(which is the < start > token) are passed to the DECODER.
- The DECODER returns the predictions and the DECODER hidden state.
- The DECODER hidden state is then passed back into the model and the predictions are used to calculate the loss. While training, we use the Teacher Forcing technique, to decide the next input of the DECODER.
- Final step is to calculate the Gradient and apply it to the optimizer and backpropagate.

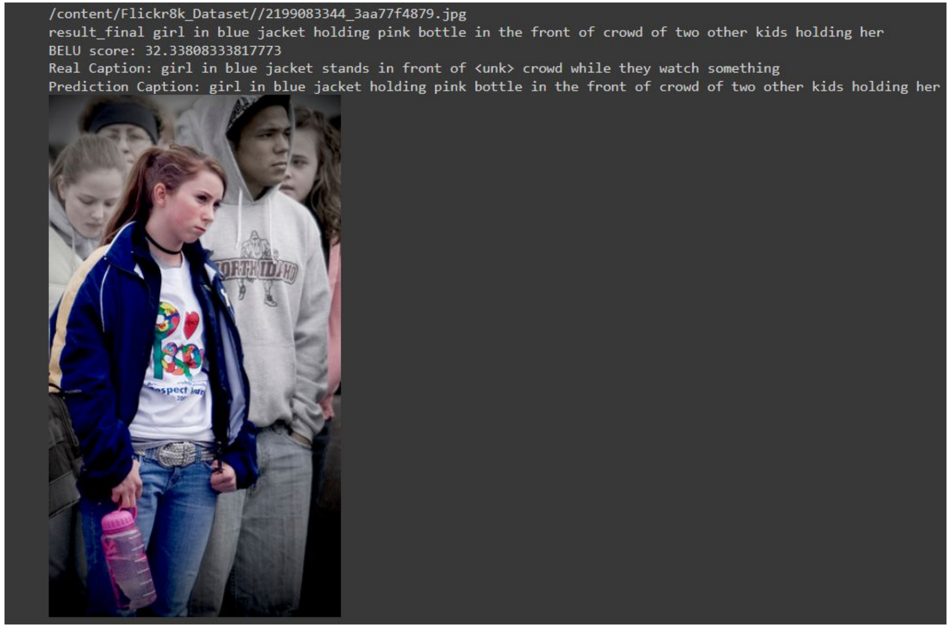
## Predict caption of the image and evaluate with bleu score:

BeamSearch is used to make prediction for the caption of the image.

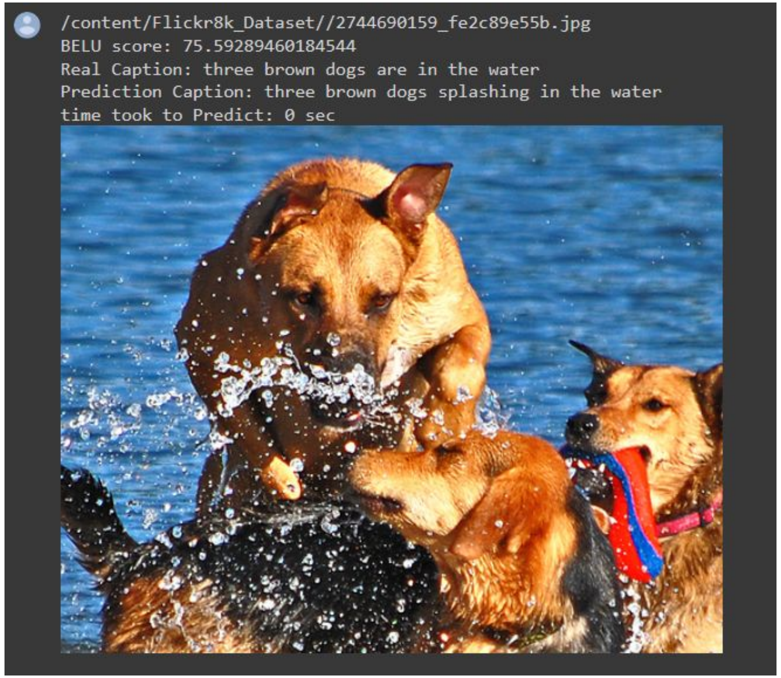
Top number of predictions are extracted, then feed them again in the model and then sort them using the probabilities returned by the model. So, the list will always contain the top k predictions. In the end, we take the one with the highest probability and go through it till we encounter <end> or reach the maximum caption length.

BLEU measure to evaluate the result of the the test set generated captions. The BLEU is simply taking the fraction of n-grams in the predicted sentence that appears in the ground-truth.

## Predict caption of the image and evaluate with bleu score:



```
/content/Flickr8k_Dataset//2199083344_3aa77f4879.jpg
result_final girl in blue jacket holding pink bottle in the front of crowd of two other kids holding her
BELU score: 32.33808333817773
Real Caption: girl in blue jacket stands in front of <unk> crowd while they watch something
Prediction Caption: girl in blue jacket holding pink bottle in the front of crowd of two other kids holding her
```



```
/content/Flickr8k_Dataset//2744690159_fe2c89e55b.jpg
BELU score: 75.59289460184544
Real Caption: three brown dogs are in the water
Prediction Caption: three brown dogs splashing in the water
time took to Predict: 0 sec
```