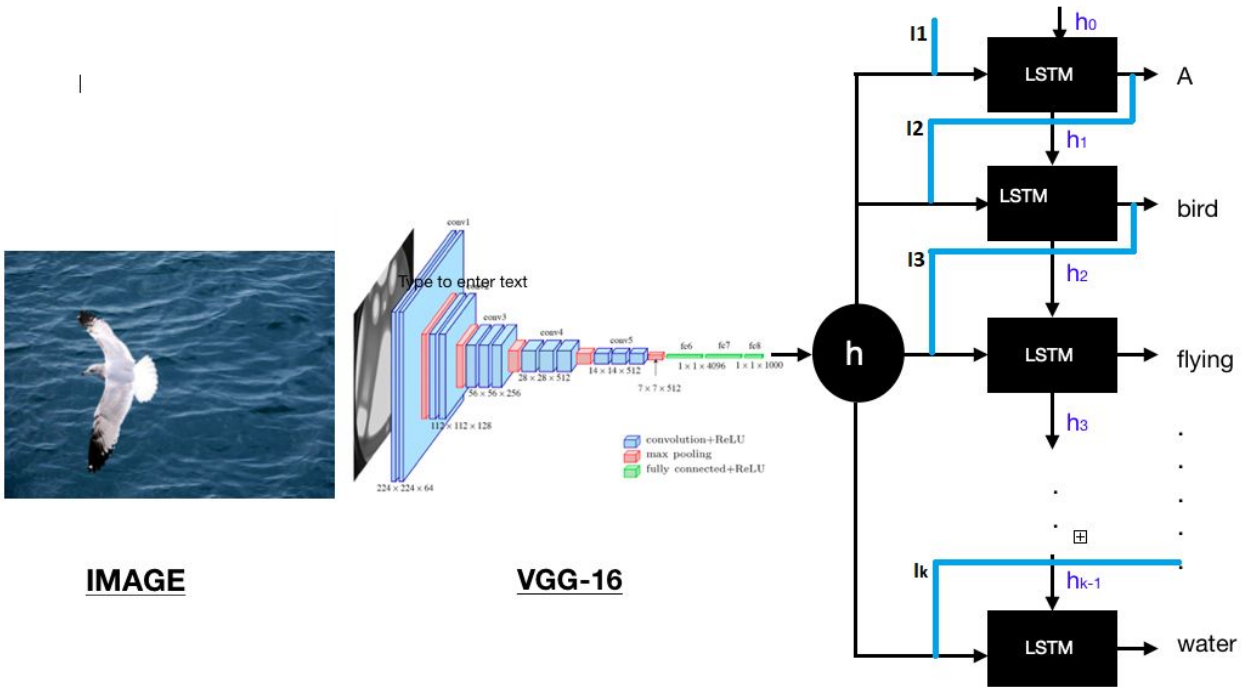
IMAGE CAPTIONING WITH VISUAL ATTENTION

Introduction

Image captioning is the process of generating textual descriptions of an image.It uses both Computer Vision to understand the content of image and language model from field of Natural Language Processing to turn the understanding of image to words .Simple/Classic Image captioning model consist of Encoder (Convolution Neural Network (CNN) model) and Decoder(Gated Recurrent Unit (GRU) or Long Short Term Memory Unit (LSTM)).In Simple/Classic image captioning model ,image is encoded with the help of pre trained Convolutional Neural Network (ENCODER) that would produce the features of image.Then ,these features are decoded by LSTM or GRU (DECODER) and generates recursively each word of the caption**.**



The problem with this model 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 the essence of the entire input image.So basically final caption won't be describing about the image instead it will give different word for different part.Because of this reason ,Attention model comes into picture. So, in this project ,we are using pretrained model inceptionV3 as an Encoder ,GRU as a Decoder ,Soft Attention model and Flickr 8k dataset.

InceptionV3:

InceptionV3 is a pretrained CNN model by Google.It is the third version in a series of deep learning convolutional architecture.Inception V3 was trained using a dataset of 1,000 classes from the original ImageNet

dataset which was trained with over 1 million training images.It is 42 layer deep with layers as convolution,AvgPool,MaxPool,concat,Fully Connected,DropOut, and softmax.It takes input of dimension 299x299x3 and second last layer with dimension 8x8x2048.

Why InceptionV3?

Reason for selection of the InceptionV3 model is that it has comparatively less number of parameters and similar complexity as VGGNet.

Attention:

Attention models, or attention mechanisms, are input processing techniques for neural networks that allows the network to focus on specific aspects of a complex input, one at a time until the entire dataset is categorized .To some extent, motivated by how we pay visual attention to different regions of an image or correlate words in one sentence.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.There are two types of attention model i.e. Soft Attention and Hard Attention.

In soft attention,the alignment weights are learned and placed “softly” over all patches in the source image,the model is smooth and differentiable but expensive when source input is large.While in hard

attention,it only select one patch of image with some probability,the model has less calculation at inference time but this is non-differentiable and require more complicated technique such as variance reduction and reinforcement learning to train.So considering the smoothness we selected soft attention model.

Framework and packages used:

Tensorflow,Scikit-learn,Matplotlib,String,Os,Time,Tqdm,Numpy,Pickle

Details:

We worked on Flickr 8k dataset which consists of 8091 images with 5 captions each.We had loaded the dataset in dictionary with image ID as key and list of 5 captions as value.Then we preprocessed the caption with cleaning by removing numbers and punctuations and added the start and end token....Why?..So that Model can understand where to start and where to stop while generating captions.Then we stored the image path and its caption in respective list with a similar index .

Then we did some data analysis like finding the total number of Unique words (9024) and their frequency ,by seeing the bar plot of top 50 and least 50 words.

We need features of Image that are feeded to decoder.These features are extracted with the help of pretrained Inceptionv3 model by taking out the learned parameters from the model.Preprocessing of Image is needed here because inceptionV3 takes input of fixed dimension i.e

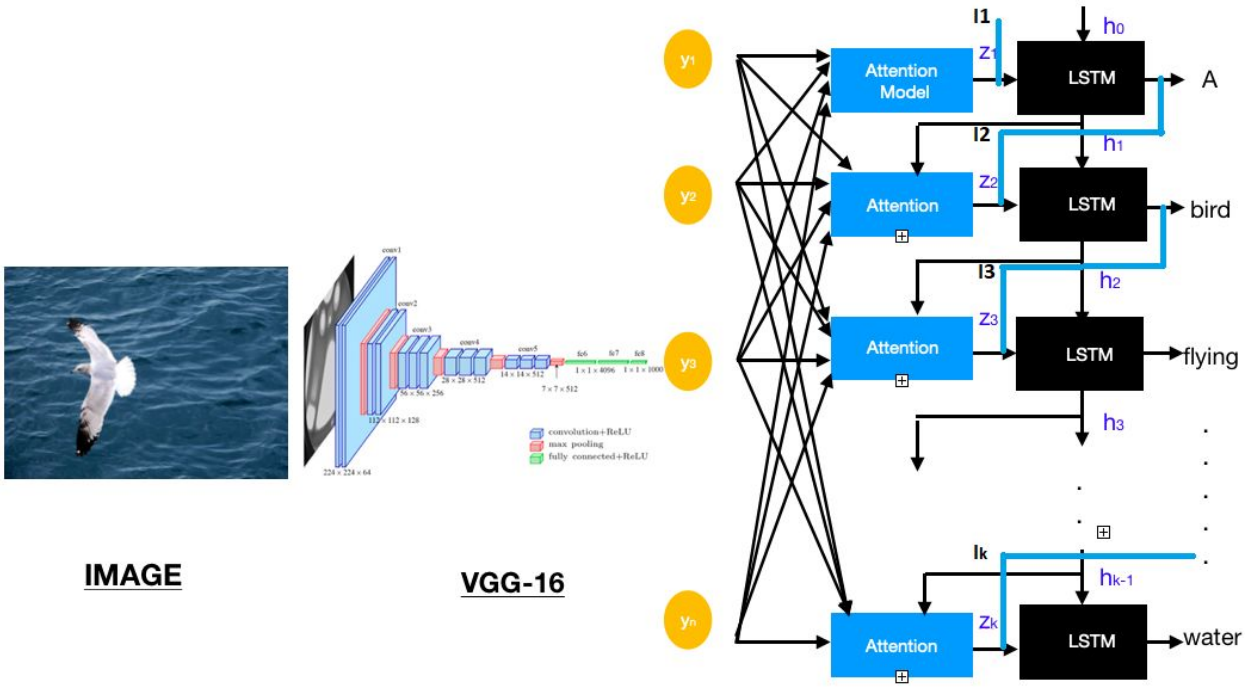
299x299x3 and image are of different sizes So converting them to required dimension.We had saved the features of all the images in its corresponding path so that it can be used while training.We did this because it takes a lot of time to extract features from the image, so ,it's better to save the feature and reuse again.

Further preprocessing is done on captions,as we need numerical input to decode so we converted all the captions in the form of numbers.This is done by making a dictionary of word_to_index and index_to_word so that we can interconvert both.We took the vocab size of 5k and replaced other words with UNK token.We selected less word to save memory and time.Since captions may be differ in length and we need to give input of same length to decoder (why?..to maintain the smoothness of the model and better performance) , so we need to do padding with max length of caption.These all work is done by tensorflow's tokenizer. Then we loaded the saved features of the images.

Model workflow:

Output of InceptionV3 model is of dimension 8x8x2048 ,where 8x8=64 represents the 64 locations/pixels of the image(**y1,y2...y64)** and 2048 are the features of those locations. After that we are converting these features to embedding dimension i.e 256 for which we used Dense layer of tensorflow.Now we have 64x256 that we are going to feed to decoder but we are using attention model so it will first get pass through attention layer (using previous hidden state $\mathbf{h}_{t-1}$(1x512) of

decoder and all features(64x256)) where it is calculating the attention weights (64x1) for all the pixel according to which it will pay attention to particular location and generating context vector $\mathbf{Z_t}$(1x256).



After this context vector is concatenated with the previous input $\mathbf{I_t}$(1x256)[for e.g starting input is <start> which is of dimension 1x1 ,it is converted into 1x256 with the help of tensoflow's embedding layer which uses some algorithm to generate embedding].

After concatenating context vector and input ,the main input (**Z+I**) to decoder is of dimension 1x512 .So there are two input to the decoder model i.e (**Z+I**) and previous hidden state **ht**. After feeding these input to GRU model we get output (length x 512) and hidden state (1x512).But we need output of vocab_size with probability of words so we use further layers (Dense) to make it to (1xvocab_size) and using softmax

function in this Dense layer for the probability.And this cycle continues till we get the <end> token.Sparse Categorical Cross Entropy Loss is used as loss function which determines that loss is getting reduced and stabilizing and Adam is used as an optimizer which updates the gradient of the variables.

With this model learns its parameters which are used for evaluation.

Evaluation method:

There are two methods :Greedy Search and Beam search.

We implemented using greedy search.In this method , we take the most probable word as a result and then feed it to the next model till we get a <last> token.

While in Beam search, we take most beam_size probable words and at last which has the highest probability will be taken as a result.

Deployment:

We made a web app which takes the input image and shows its generated captions .Then deployed it using Heroku.

Deployment file requirement:

- Main App.py where main evaluation is done
- Procfile which contains the start of the app. This file will tell the heroku to start this particular app.

- Requirements.txt which will tell heroku to install all these files in the app

Github link: Image Captioning code

App link: Demo