Image Captioning using Attention Model vs Transformer

# **Summary**

Image Captioning refers to describing the information contained in an image in textual form. In this Image Captioning Project, we applied two different Deep Learning techniques for image captioning: Attention Models and Transformers, selected an evaluation metric, compared the results of both and found Transformers to be doing a better job than Attention Models.

Justification of Transformers performing better was explained in detail. The conclusion was drawn that Transformers perform better. Finally, future improvements were suggested.

# **Problem Statement**

Image Captioning is a task where we define a caption for an image. Caption means a sentence which is used to describe the image. Image Captioning is an increasingly popular problem in the field of Deep Learning where we leverage the capability of deep neural networks to extract features from the data. Two such techniques are Attention Models and Transformers. We applied both in this project and then compared the results.

The Dataset we used for this project is from Kaggle. It is called FLICKR8K Dataset and contains roughly 8000 images, along with 5 captions for each image.

Both techniques which we used are explained in detail below.

# **Techniques**

### **Attention Model:**

The first technique we used to solve the Image Captioning problem was Attention Models.

Attention Models are a Deep Learning Neural Network based technique where the model learns by having an additional attention or focus on each component of a sequence. A sequence here could be a sentence or an image. In this case, the sequence is an image. More specifically, a vectorized representation of an image.

Attention Models have an encoder-decoder architecture. In Attention Models, we pass the input sequence sequentially, which means passing each word or a sentence or part of an image, one by one, through the encoder layer, where the component’s encoding is done by trying to learn the relative importance of all the other components on the component. This is called self-attention and it helps the model learn long term context information better than LSTMs and RNNs.

To sum up, attention models are neural networks that focus on specific aspects of a complex sequential input, one at a time until the entire dataset is categorized. Since they focus on one component of the input sequence at a time, it makes them more efficient than the standard LSTMs, particularly for longer sequences.

In our project, we used Attention Model for Image Captioning. The architecture we used had the below structure:

Encoder: A pertained Convolutional Neural Network (CNN) known as VGG-16 (The name comes from the fact that it has 16 layers)

Decoder: A Recurrent Neural Network

The purpose of the Encoder here was to extract important features from the image. Since we know that Convolutional Neural Networks are the most popular choice for dealing with Image Data, since they reduce the dimensionality of image without losing its information, we used a pertained Convolutional Neural Network (CNN), VGG-16 for our task. Below figure shows the VGG-16 architecture.

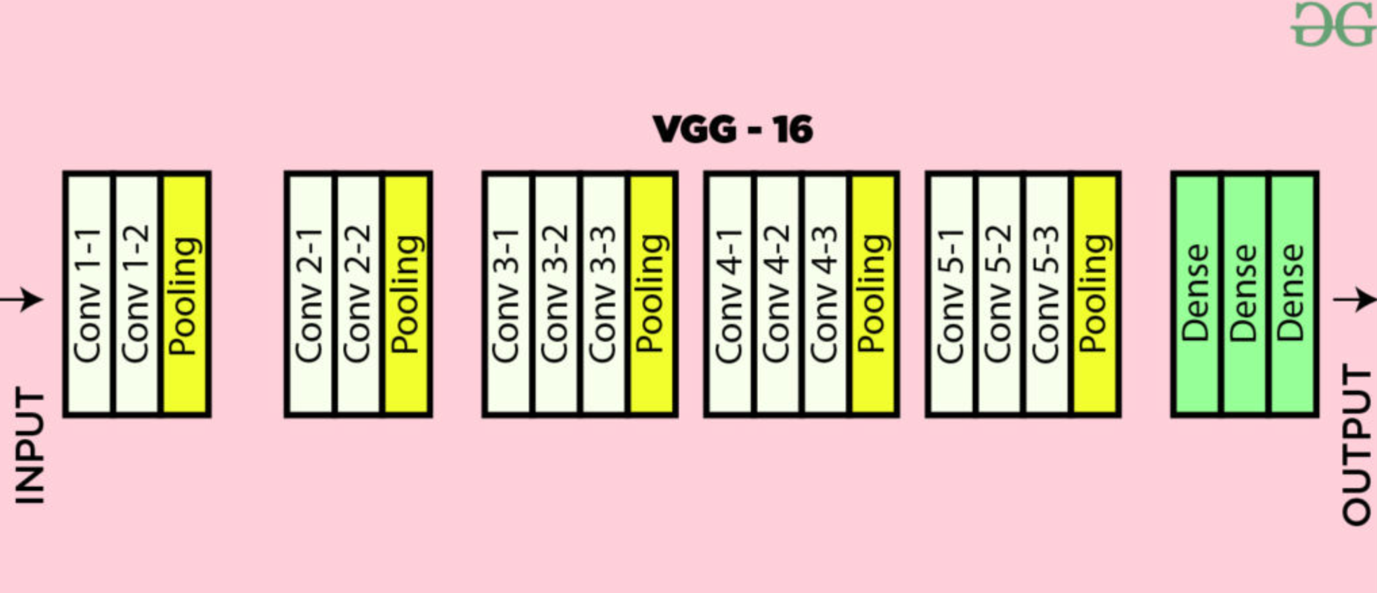


Figure 1: VGG-16 Convolutional Neural Network

As shown in the figure 1 it has several convolutional layers, as well as some pooling layers which help in reducing the number of training parameters. The dense layers are not being used in our cases since we only need the learned features (vectorized representation) of our input image, and not trying to classify anything,

The output of the Encoder is then decoded with the help of a Recurrent Neural Network, together with attention mechanism to generate the image caption. Below picture gives an idea of how the attention model works on the Decoder side of things:

Diagram, engineering drawing

Description automatically generated

Figure 2: Attention Mechanism

In figure 2, the different x components are components of an image, for each of which attention is calculated by also considering the previous hidden state as well as other components.

The code was implemented using TensorFlow and Keras in Python Notebook. Below is an example of a generated caption on an image.

BELU score: 4.677477553605556e-76

Real Caption: brown dog jumps high on field of grass

Prediction Caption: brown dog jumps into the grass



Figure 3: Word by Word Caption Generation



Figure 4: Original Image

## **Transformers:**

The Transformer Neural Network is a neural network-based architecture that aims to solve sequence-to-sequence tasks while handling long-range dependencies with ease. Transformers are sed in image captioning, machine translation etc.

Like Attention Models, Transformers also have an Encoder Decoder architecture, with some additional components. Below image shows Transformers Architecture.

Diagram

Description automatically generated

Figure 5: Transformers Architecture

Transformers take all input words or the entire image at once. Unlike the sequential approach of LSTMs/RNNs or attention models, input image is then divided into smaller components. The components of the image, which are basically partial images, need sequence information since each component or image part need to know where it should go. Sequence information is provided via positional embeddings in Transformers, as shown in the above Transformers Architecture image.

The output from positional embeddings is fed into a multiheaded attention layer in parallel.

By parallel we mean all image parts at once. Multi headed attention is a component in the Transformers architecture that computes self-attention for each image part in parallel several times (heads). In each head, the model learns a different contextual information about the input image.

It is important to note that by self-attention we mean creating the encoded representation of each image part in context of all other image parts. The output is then aggregated and normalized and passed through a feed forward layer.

For decoder block, the output of previous sequence (image part) is fed to another multi headed attention layer. The output of that is then aggregated, normalized, and given together with encoder block output to another multi headed attention layer.

The output of that is then again aggregated, normalized, and provided to a feed forward layer.

Finally, a SoftMax layer is used to predict the caption for that image part (for example an image part may contain cat, while full image is cat playing with a ball)

The code was implemented using TensorFlow and Keras in Python Notebook. Below is an example of a generated caption on an image using Transformers.

BLEU-1 score: 23.529411764705884

BLEU-2 score: 7.235653323902229e-153

BLEU-3 score: 1.6591665362080695e-183

BLEU-4 score: 1.268852357850863e-229

Real Caption: Basketball players leap for the ball while the crowd watches

Predicted Caption: Basketball player in white uniform is jumping up into the air while another player in white watches



Figure 6: Original Image

## **Code Execution:**

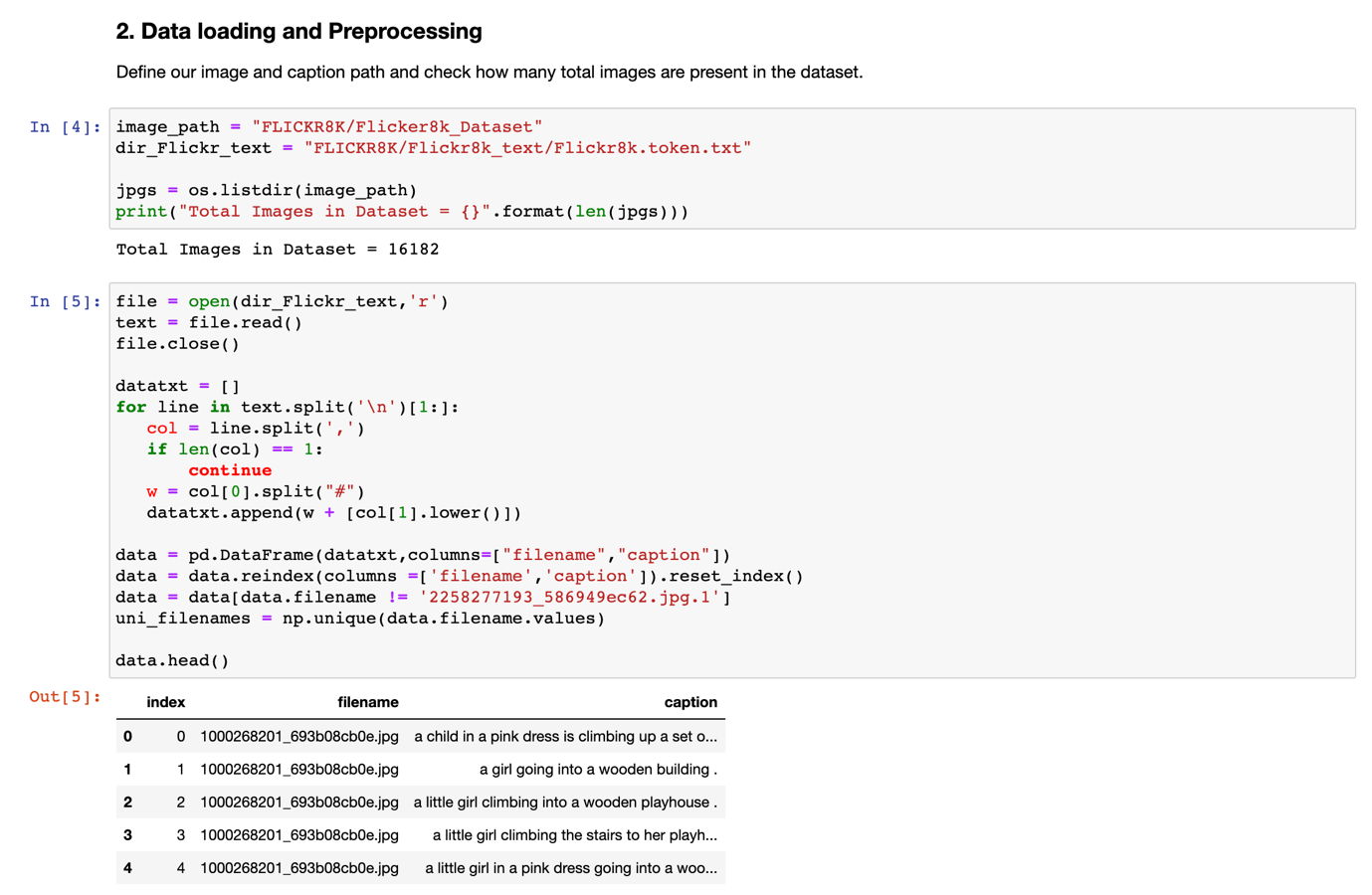
Transformer Model

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated



Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text

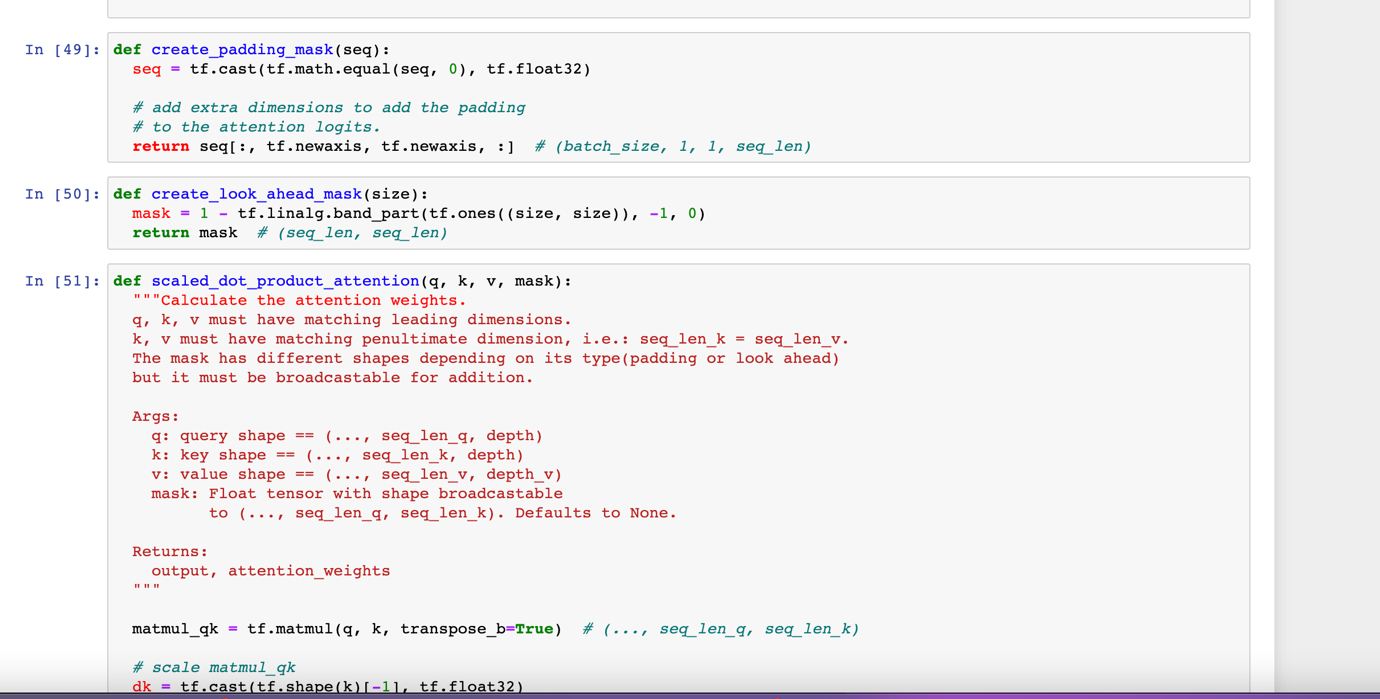
Description automatically generated

Graphical user interface, text, application, email

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generatedGraphical user interface, text

Description automatically generatedGraphical user interface, text, application

Description automatically generatedGraphical user interface, text

Description automatically generated

# **Contribution**

I used a VGG-16 CNN in Attention Models as our encoder to extract features from the images. I used it little differently than normal since we excluded the last SoftMax layer of VGG-16. This was done to extract only the image vectors, and not to classify anything since we needed to use this network as an encoder. Below code shows this:

Table

Description automatically generated

Figure 9: VGG-16 Architecture

In above picture, we can see that in the first line where we load the VGG-16 Network where the weights parameter is ImageNet which means this CNN network is pretrained on ImageNet dataset. In the third line, we omit the last SoftMax layer of the VGG-16 CNN network and then create our model using the input and all other hidden layers of the network. The output displayed shows some of the layers of the network, which are either convolutional layers or pooling layers.

We tweaked the hyper parameters of the CNN network we used in Attention Models which improved the performance. The final values of the hyper parameters are shown below.

Graphical user interface, text, application, email

Description automatically generated

Figure 10: Hyper Parameters for the VGG-16 Encoder

In above picture the values we used were obtaining via hit and trial, by testing the model out on various values of batch size, hidden units and so on.

## **Results Comparison**

Let’s compare the BLEU scores achieved using Attention Model vs Transformers for the same image.

BLEU score helps to compare the generated image captions with the provided reference captions since multiple captions can be good. It uses modified precision on n-grams in the generated image caption compared to the provided reference captions. BLEU also has an adjustment factor to penalize shorter image captions than the reference image caption

The typical value for BLEU Score is in the range of 0-1 or 0-100

Now, let’s compare the BLEU scores achieved using Attention Model vs Transformers for the same image. Top pic is Attention Model result while bottom pic is Transformers.

Graphical user interface

Description automatically generated with medium confidence

Figure 7: Attention Model Results

Graphical user interface

Description automatically generated

Figure 8: Transformers Model Results

Clearly Transformers result is better and has higher BLUE score (68.65 vs 30.04 of Attention Model)

We saw that Transformers performed better but why is that?

In Attention Model, we used a pretrained convolution neural network VGG-16 (encoder) followed by an RNN based decoder. CNNs use filters on parts of an image to extract useful features and then the RNN generates image caption by processing each sequence of the image one at a time (also using input from last hidden state)

While Transformers use the entire image at once which helps it learn better. Now that makes sense but let us explain that intuition further. In Transformers, the whole image is fed at once. Image is then split into patches which are then provided sequence information (which patch should go where) based on positional embeddings.

The encoded output from positional embeddings (image patches with sequence information) are then fed in parallel to a Multi Headed Attention layer. The Multi headed attention layer is one main reason why transformers perform better since it helps the model to learn the long-term context information better and in a richer way.

# **Conclusion**

To conclude, we applied two different techniques for Image Captioning, both of which were Deep Learning based methods. Transformers performed better than Attention Models based on our evaluation metric, BLEU score, mainly because Transformers process the entire image at once and use multi headed attention layers. For future improvements, we can try to speed up the Transformers network by adding some dropout layers or some sort of regularization.

## **References**

American express (2022) *Attention mechanism in deep learning: Attention model keras*, *Analytics Vidhya*. Available at: <https://www.analyticsvidhya.com/blog/2019/11/comprehensive-guide-attention-mechanism-deep-learning/>

Kulshrestha, R. (2020) *Transformers*, *Medium*. Towards Data Science. Available at: <https://towardsdatascience.com/transformers-89034557de14>

Dataset: <https://www.kaggle.com/datasets/adityajn105/flickr8k>