Visual Recognition Final Project

Jainav Sanghvi — IMT2020098

Instructor: Prof. Viswanath G

Date: September 27, 2023

1 Problem Statement

Our objective is to create a CNN-LSTM model for generating captions for images using the Flickr8 dataset. Initially, we will develop a basic model with a chosen CNN-LSTM architecture and evaluate its performance. We will then enhance both the vision and language embeddings to improve upon the baseline model, without making any changes to the LSTM's architecture.

Afterwards, we will compare the modified baseline with the original using different metrics such as BLEU and METEOR scores to determine the effectiveness of the improvements. Our goal is to create an accurate and efficient image captioning system that can produce natural and descriptive captions for various types of images.

2 Dataset

For the image caption generator, we will be using the Flickr_8K dataset. There are also other big datasets like Flickr_30K and MSCOCO dataset but it can take weeks just to train the network so we will be using a small Flickr8k dataset. The advantage of a huge dataset is that we can build better models.

3 CNN-LSTM System

A CNN-LSTM system is a neural network architecture that combines two powerful deep learning models: Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks.

CNNs are good at extracting spatial features from images, while LSTMs excel at capturing the temporal sequence of language data. By combining these two models, a CNN-LSTM system can effectively generate captions for images by leveraging the visual and textual information.

In the case of image captioning, the CNN part of the model is used to extract high-level features from the input image, while the LSTM part of the model processes these features to generate a sequence of words that describe the image. The LSTM takes in the output of the CNN and the previous word in the sequence, and generates the next word in the sequence until the entire caption is generated.

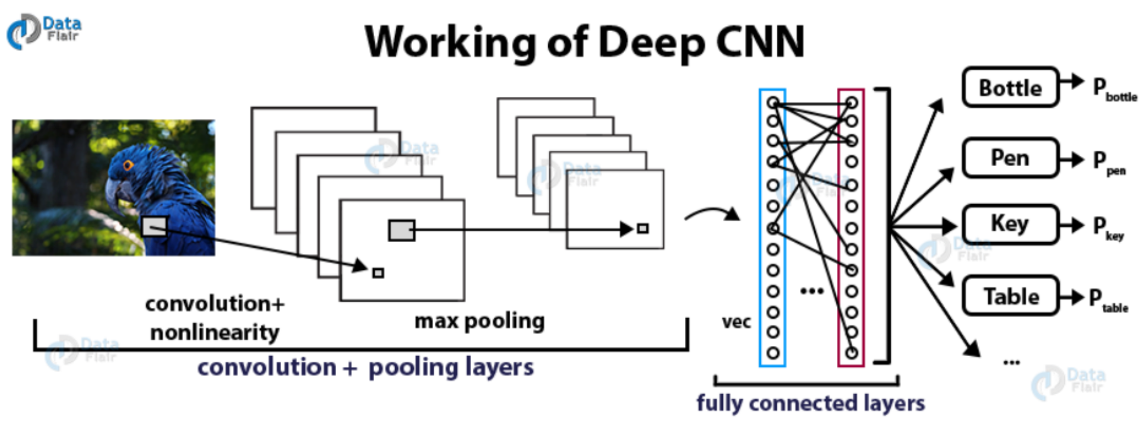
Overall, the CNN-LSTM model is a powerful approach for generating image captions that are both accurate and descriptive, and it has been successfully applied in many computer vision tasks, including object detection, image segmentation, and image classification.

3.1 CNN

Convolutional Neural networks are specialized deep neural networks which can process the data that has input shape like a 2D matrix. Images are easily represented as a 2D matrix and CNN is very useful in working with images.

CNN is basically used for image classifications and identifying if an image is a bird, a plane or Superman, etc.

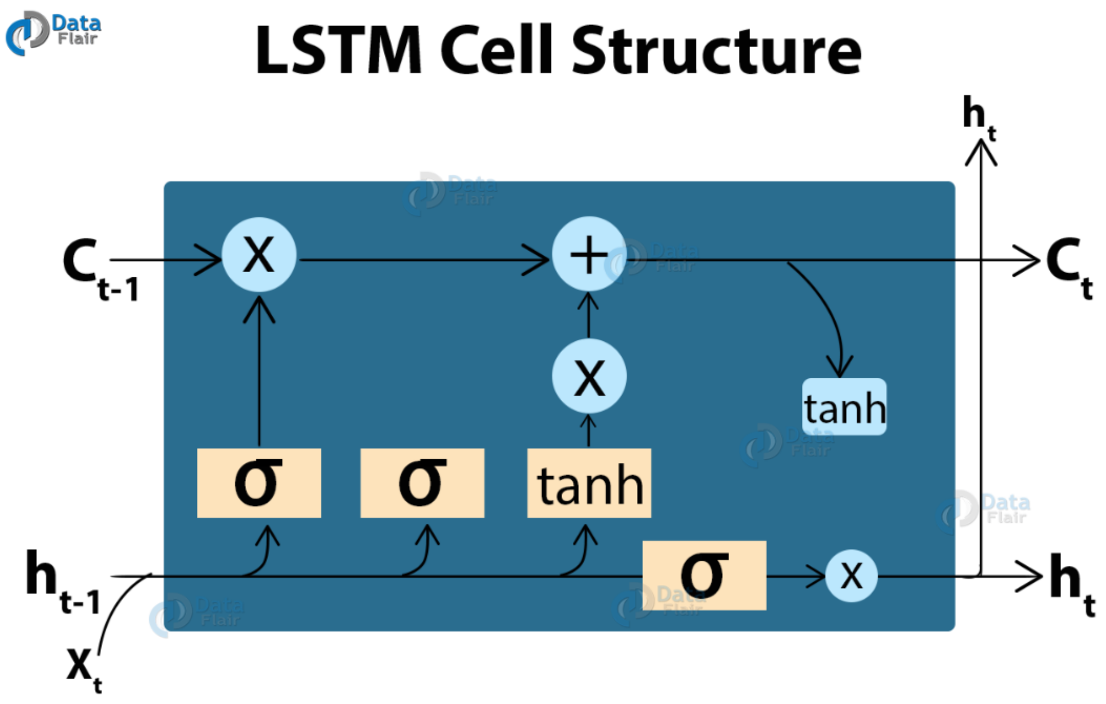
It scans images from left to right and top to bottom to pull out important features from the image and combines the feature to classify images. It can handle the images that have been translated, rotated, scaled and changes in perspective.



3.2 LSTM

LSTM stands for Long short term memory, they are a type of RNN (recurrent neural network) which is well suited for sequence prediction problems. Based on the previous text, we can predict what the next word will be. It has proven itself effective from the traditional RNN by overcoming the limitations of RNN which had short term memory. LSTM can carry out relevant information throughout the processing of inputs and with a forget gate, it discards non-relevant information.

This is what an LSTM cell looks like –



3.3 Applications

The combination of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) models has proven to be effective in various applications, including vision and language tasks such as visual question answering and image caption generation. Let's explore these applications in more detail:

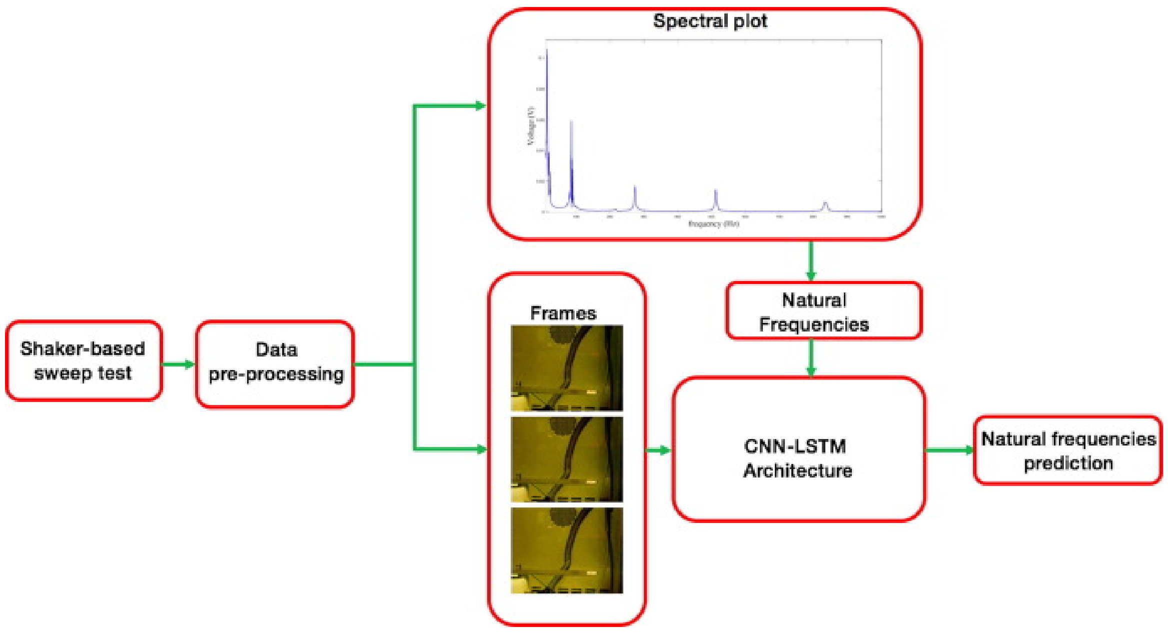
3.3.1 System For Vision + Language

CNN + LSTM for Vision + Language is a type of neural network architecture that combines the strengths of both CNNs and LSTMs to perform tasks that require understanding of both visual and textual information.

In this architecture, the CNN is used to extract visual features from the input image, while the LSTM is used to process the language data. The visual features from the CNN are fed as input to the LSTM along with a sequence of words, which are embedded as vectors in a high-dimensional space.

The LSTM then processes the sequence of word embeddings along with the visual features from the CNN, and generates the output in the form of a sequence of words that form a natural language sentence. The LSTM has a memory that allows it to store information about previous words in the sequence, which helps it generate more accurate and coherent sentences.

The CNN + LSTM for Vision + Language architecture is widely used in many applications such as image captioning, visual question answering, and image-text matching. The ability to combine visual and textual information makes this architecture particularly useful in tasks where understanding both modalities is important.



A typical CNN-LSTM System for Vision + Language

3.3.2 System For Image Captioning

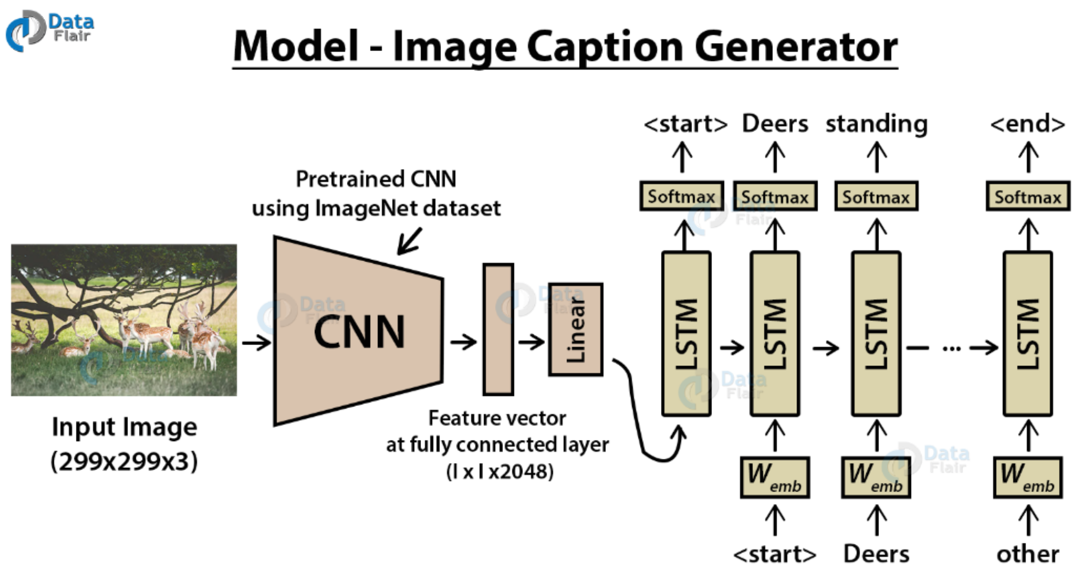
CNN + LSTM architecture is widely used for image captioning. The basic idea of this architecture is to use a CNN to extract visual features from an image and use those features as input to an LSTM that generates a corresponding caption.

In this architecture, the CNN is typically pre-trained on a large dataset such as ImageNet to extract high-level features from the image. The output of the CNN is a fixed-length feature vector that represents the visual content of the image.

The LSTM then takes the visual features as input and generates a sequence of words that describe the image. The LSTM uses its memory to keep track of previously generated words and their context, allowing it to generate a coherent and meaningful caption.

The training process for this architecture involves feeding the model with image-caption pairs, and adjusting the model's parameters to minimize the difference between the predicted captions and the ground-truth captions.

Overall, the CNN + LSTM architecture for image captioning has shown promising results and has been widely used in various applications such as image description, visual storytelling, and image retrieval.



4 Methodology

In order to develop a robust CNN-LSTM model for image captioning using the Flickr8k dataset, several steps were taken to preprocess the dataset. The captions associated with each image were separated into a dictionary with the image names as keys and their corresponding captions as values. The dataset was then split into three separate dictionaries for training, testing, and validation purposes. To improve the performance of the model, sequence tokens were added to the captions to mark the beginning and end of the caption. The maximum length of the captions was computed to ensure that every caption was of constant length for better computations. Finally, the vocabulary size of the dataset was calculated to determine the number of unique words present in all the captions, which was necessary for computing the word embedding matrix. These preprocessing steps were crucial for building an accurate and efficient image captioning system.

Step 1: **Preprocessing**

(a) **Setting up captions:**
For each image in the Flickr8k dataset, there are five captions associated with it. To separate the image names from their captions, a dictionary was created with the image names as keys and their corresponding five captions as the values in the form of a list.

(b) **Splitting into Train, Test and Validation:**
To create separate datasets for training, testing, and validation purposes, three dictionaries were created by copying the originally created dictionary into them.

(c) **Adding sequence tokens:**
Sequence tokens, such as start and end tokens, were added to the text to mark the beginning and end of the caption. A "startcap" token was added as a start token and an "endcap" token as an end token to each caption.

(d) **Caption maximum length computation:**
The maximum length of the caption was calculated to ensure that every caption is of constant length for better computations and to prevent excessively long caption generation. This was done by iterating through the captions and updating the maximum value whenever a caption size greater than the current maximum was encountered.

(e) **Calculating vocabulary size:**
The vocabulary size, which is the number of unique words present in all the captions in the dataset, was computed. This was done by iterating through the captions and adding every unique word to a list called vocab_list. The vocabulary size was determined by the length of the vocab_list.

Step 2: **Word Embeddings Generation**

(a) **Obtain GloVe Embeddings:**
First, acquire the GloVe embedding file that contains pre-trained word embeddings. Each line in the file represents a word followed by its corresponding embedding vector.

(b) **Read the GloVe Embedding File:**
Open the GloVe embedding file and iterate through each line. Split each line to extract the word and its embedding vector.

(c) **Create the *glove_emb_dict* Dictionary**
Initialize an empty dictionary called glove_emb_dict. For each word and its embedding vector in the GloVe embedding file, add the word as the key and the embedding vector as the value to the glove_emb_dict dictionary.

Step 3: **Create the Embedding Matrix**

(a) **Prepare the Vocabulary List:**
Create a list called vocab_list that contains all the words from your own vocabulary. Ensure that the words in vocab_list are in the same order as they appear in your training data.

(b) **Create the Embedding Matrix:**
Initialize an embedding matrix with dimensions (vocab_size, embedding_dim) using a random initialization method such as np.random.uniform. Iterate through each word in vocab_list and check if the word exists in glove_emb_dict. If the word is present, retrieve its corresponding embedding vector from glove_emb_dict. Assign the embedding vector to the corresponding row in the embedding matrix based on the word's index in vocab_list. If the word is not present, you can choose to initialize the embedding randomly or assign a special token embedding for out-of-vocabulary words.

The subsequent sections outline the next set of steps, excluding commonplace techniques like data loading and testing, as they are widely known and do not directly contribute to the core concept of the project.

For Training, the Hyperparameters are:

- **Number of Epochs**: Refers to the number of times the learning algorithm will work through the entire training dataset. Each epoch consists of one forward pass and one backward pass of all the training samples. Increasing the number of epochs allows the model to see the training data more times, potentially improving its accuracy. However, a large number of epochs can lead to overfitting if the model starts memorizing the training data instead of generalizing from it. **This is set to 15 for our case.**
- **Batch Size**: Represents the number of training samples shown to the model before the model's internal parameters are updated. In other words, it determines the number of samples that are processed together in parallel during each training step. The batch size affects the speed and stability of the training process. Smaller batch sizes allow for more frequent weight updates and may result in faster convergence. However, very small batch sizes can lead to noisy gradients and slower convergence. Larger batch sizes can provide more stable gradients but might require more memory to store intermediate activations. **This is set to 64 for our case.**

5 Model Evaluation Metrics

In the context of Image Captioning, Meteor and BLEU (Bilingual Evaluation Understudy) are two commonly used evaluation metrics to assess the quality of generated image captions by comparing them to reference captions.

Both Meteor and BLEU are automatic evaluation metrics, meaning they provide a numerical score to measure the quality of generated captions automatically, without human involvement. These metrics serve as proxies for assessing how well the generated captions match the reference captions in terms of linguistic quality, fluency, and relevance to the corresponding image.

5.1 BLEU Score

BLEU is a popular metric for evaluating the quality of machine-generated translations or text generation tasks, including image captioning. It compares the n-grams (contiguous sequences of n words) between the generated caption and the reference captions. BLEU computes a precision-based score that ranges from 0 to 1, with higher scores indicating better quality captions.

5.2 Meteor Score

Meteor is a metric that measures the quality of machine-generated text by considering both precision and recall. It takes into account the unigram, bigram, and trigram matches between the generated caption and the reference captions. Meteor also considers the order of words in calculating the score. The metric produces a score between 0 and 1, with higher scores indicating better quality captions.

6 Architecture

6.1 Extract Image Features

ResNet-50, a deep convolutional neural network architecture, is utilized as a feature extractor. Firstly, the input images are preprocessed by resizing and normalizing them to meet the requirements of the ResNet-50 model. The preprocessed images are then fed into ResNet-50, and the output is obtained from the final convolutional layer. These output features capture high-level visual representations of the images, which encode important visual information.

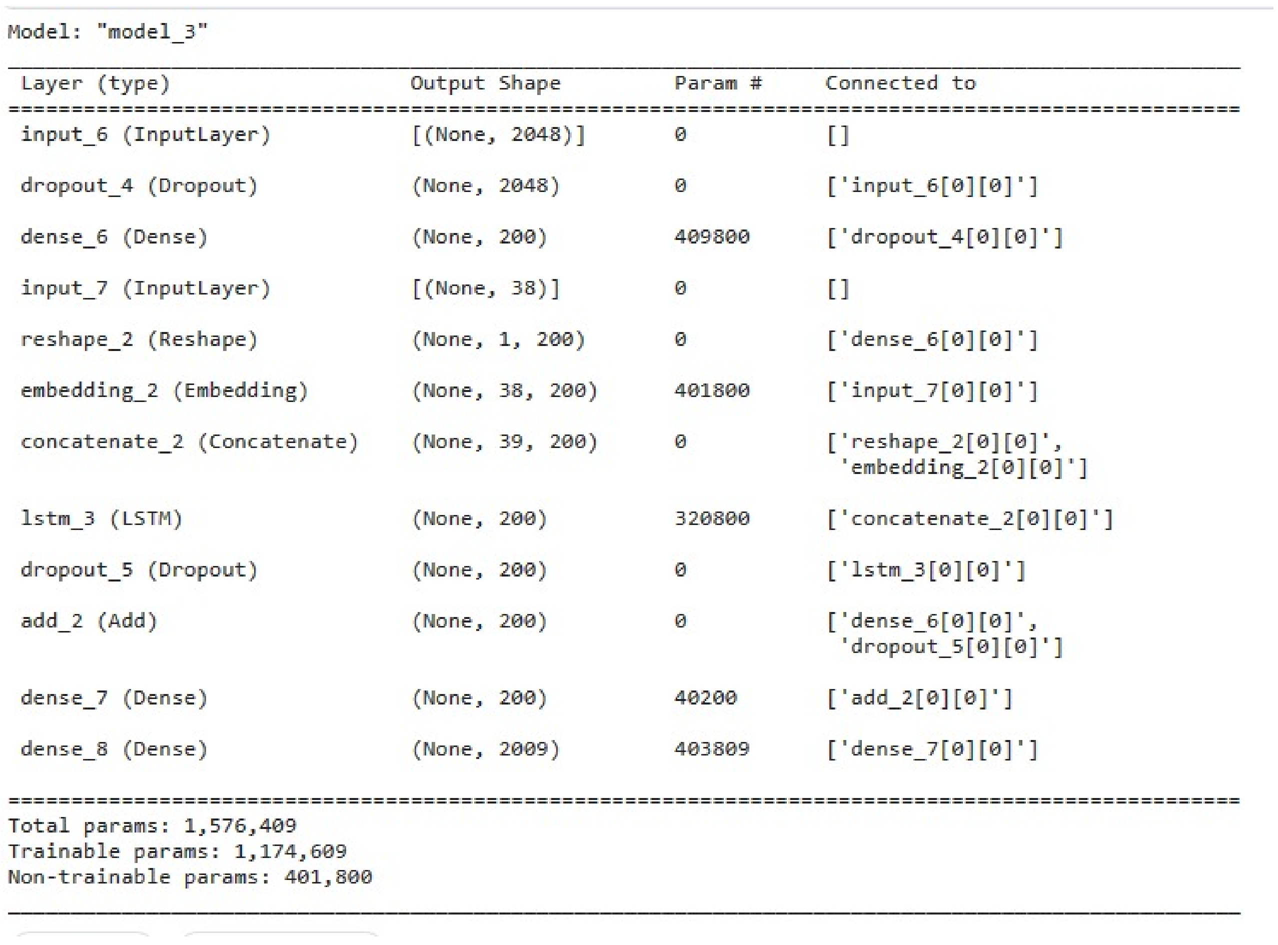
These extracted image features from ResNet-50 act as the input for the subsequent steps. The image features are fed into an LSTM network, which is responsible for capturing the temporal dependencies within the visual information. The LSTM generates a sequence of hidden states, which serves as a condensed representation of the input image features.

Finally, a fully connected layer is connected to the output of the LSTM network. This layer maps the hidden states to the vocabulary size, enabling the generation of captions. During training, the model is trained to generate captions that match the ground truth captions for a given image, using appropriate loss functions. During inference, the trained model can generate captions by predicting the next word based on the learned connections between image features and corresponding textual descriptions.

THE NEXT SUBSECTIONS WILL FOCUS ON THE ARCHITECTURES THAT WE HAVE USED AND THE COMPARISON WITH SAMPLE-GENERATED OUTPUTS.

6.2 Baseline Model

6.2.1 Summary of the Baselined Model



```
Model: "model_3"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_6 (InputLayer)           [(None, 2048)]       0           []

 dropout_4 (Dropout)            (None, 2048)         0           ['input_6[0][0]']

 dense_6 (Dense)                (None, 200)          409800      ['dropout_4[0][0]']

 input_7 (InputLayer)           [(None, 38)]         0           []

 reshape_2 (Reshape)            (None, 1, 200)       0           ['dense_6[0][0]']

 embedding_2 (Embedding)        (None, 38, 200)      401800      ['input_7[0][0]']

 concatenate_2 (Concatenate)    (None, 39, 200)      0           ['reshape_2[0][0]',
                                                                  'embedding_2[0][0]']

 lstm_3 (LSTM)                  (None, 200)          320800      ['concatenate_2[0][0]']

 dropout_5 (Dropout)            (None, 200)          0           ['lstm_3[0][0]']

 add_2 (Add)                    (None, 200)          0           ['dense_6[0][0]',
                                                                  'dropout_5[0][0]']

 dense_7 (Dense)                (None, 200)          40200       ['add_2[0][0]']

 dense_8 (Dense)                (None, 2009)         403809      ['dense_7[0][0]']

==================================================================================================
Total params: 1,576,409
Trainable params: 1,174,609
Non-trainable params: 401,800
__________________________________________________________________________________________________
```

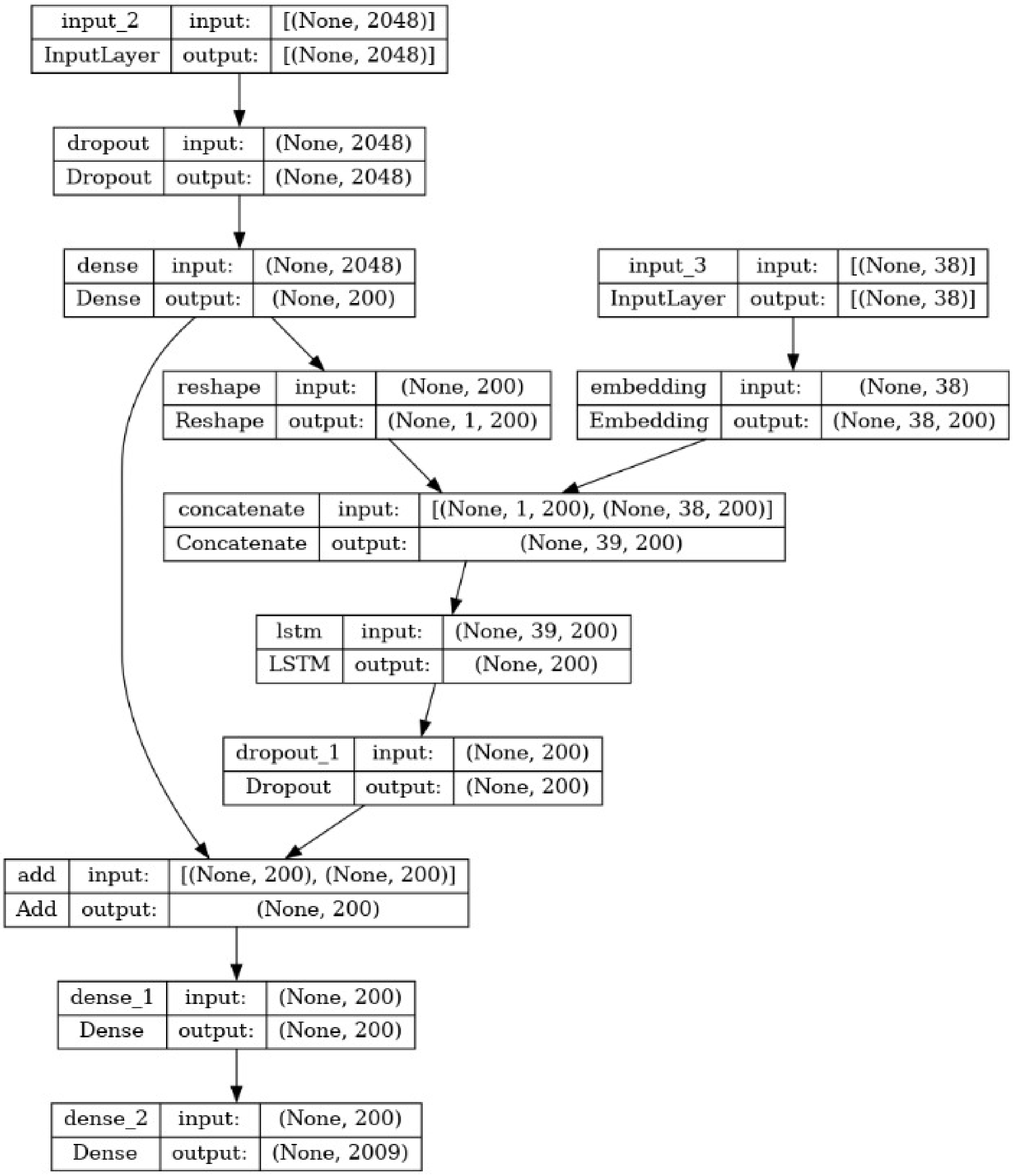
6.2.2 Model Scores

We include the maximum and average scores of both BLEU and METEOR metrics to facilitate a more comprehensive comparison between the modified model, which incorporates multi-head attention.

	Averaged Score	Maximal Score
BLEU Score	0.5757	1.0
METEOR Score	0.3464	0.8947

6.2.3 Plot Baselined Model

[29...


input_2 | InputLayer | input: [(None, 2048)] | output: [(None, 2048)]
dropout | Dropout | input: (None, 2048) | output: (None, 2048)
dense | Dense | input: (None, 2048) | output: (None, 200)
input_3 | InputLayer | input: [(None, 38)] | output: [(None, 38)]
reshape | Reshape | input: (None, 200) | output: (None, 1, 200)
embedding | Embedding | input: (None, 38) | output: (None, 38, 200)
concatenate | Concatenate | input: [(None, 1, 200), (None, 38, 200)] | output: (None, 39, 200)
lstm | LSTM | input: (None, 39, 200) | output: (None, 200)
dropout_1 | Dropout | input: (None, 200) | output: (None, 200)
add | Add | input: [(None, 200), (None, 200)] | output: (None, 200)
dense_1 | Dense | input: (None, 200) | output: (None, 200)
dense_2 | Dense | input: (None, 200) | output: (None, 2009)

6.2.4 Predictions Using Baselined Model

Prediction-1: two dog be play in the snow.

Training Sample-1 startseq a big white dog and a small black dog sit in the snow endseq.

Training Sample-2 startseq a large white dog watch a little black dog run in the snow endseq.

Training Sample-3 startseq a small dog and a large dog in the snow endseq.

Training Sample-4 startseq a tall white dog sit and watch a small black dog run around him in the snow endseq.

Training Sample-5 startseq a white dog look at a black puppy in the snow endseq

```
two dog be play in the snow
['startseq a big white dog and a small black dog sit in the snow endseq', 's
e dog sit and watch a small black dog run around him in the snow endseq', 's
```

Prediction-2: a basketball player in a white uniform be play basketball

Training Sample-1 startseq a player from white and green highschool team dribble down court defend by a player from the other team endseq.

Training Sample-2 startseq four basketball player in action endseq.

Training Sample-3 startseq four man play basketball two from each team endseq

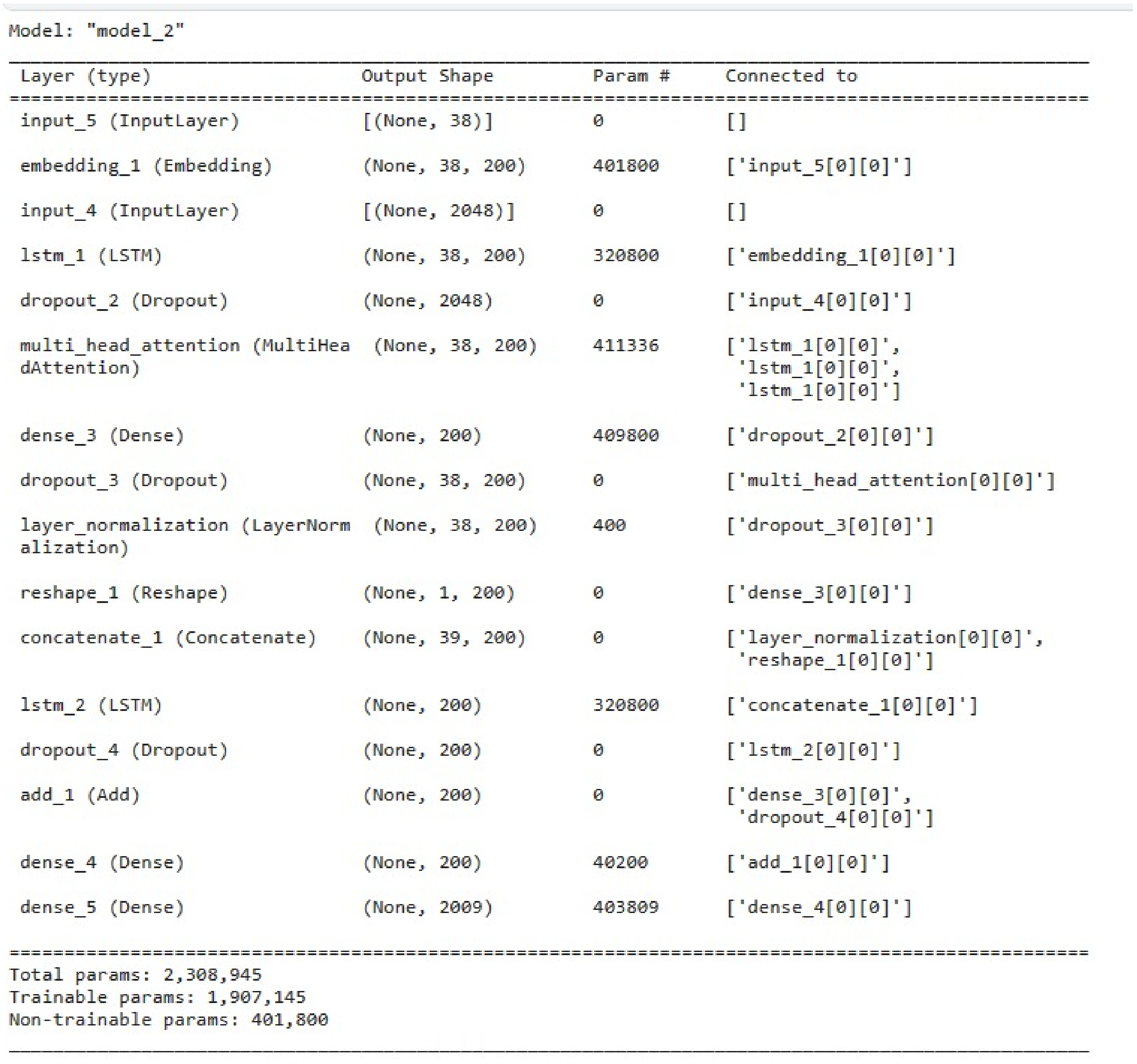
Training Sample-4 startseq two boy in green and white uniform play basketball with two boy in blue and white uniform endseq

Training Sample-5 startseq young man play basketball in a competition endseq

```
a basketball player in a white uniform be play basketball
['startseq a player from white and green highschool team dribble down court de
deq', 'startseq two boy in green and white uniform play basketball with two b
```

6.3 Modified Baseline Model

6.3.1 Summary of the Modified Baselined Model



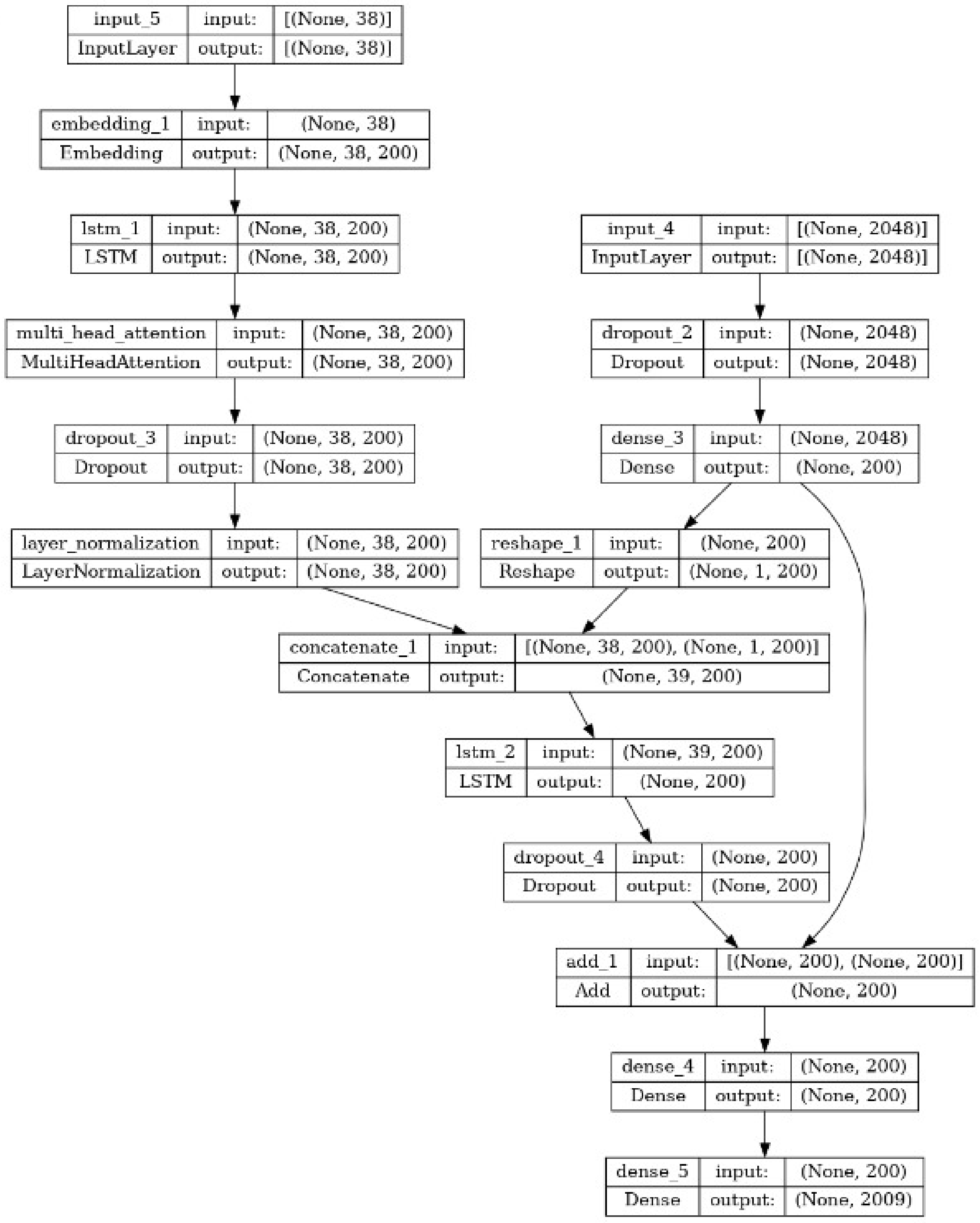
```
Model: "model_2"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_5 (InputLayer)           [(None, 38)]         0           []

 embedding_1 (Embedding)        (None, 38, 200)      401800      ['input_5[0][0]']

 input_4 (InputLayer)           [(None, 2048)]       0           []

 lstm_1 (LSTM)                  (None, 38, 200)      320800      ['embedding_1[0][0]']

 dropout_2 (Dropout)            (None, 2048)         0           ['input_4[0][0]']

 multi_head_attention (MultiHea  (None, 38, 200)     411336      ['lstm_1[0][0]',
 dAttention)                                                      'lstm_1[0][0]',
                                                                  'lstm_1[0][0]']

 dense_3 (Dense)                (None, 200)          409800      ['dropout_2[0][0]']

 dropout_3 (Dropout)            (None, 38, 200)      0           ['multi_head_attention[0][0]']

 layer_normalization (LayerNorm  (None, 38, 200)     400         ['dropout_3[0][0]']
 alization)

 reshape_1 (Reshape)            (None, 1, 200)       0           ['dense_3[0][0]']

 concatenate_1 (Concatenate)    (None, 39, 200)      0           ['layer_normalization[0][0]',
                                                                  'reshape_1[0][0]']

 lstm_2 (LSTM)                  (None, 200)          320800      ['concatenate_1[0][0]']

 dropout_4 (Dropout)            (None, 200)          0           ['lstm_2[0][0]']

 add_1 (Add)                    (None, 200)          0           ['dense_3[0][0]',
                                                                  'dropout_4[0][0]']

 dense_4 (Dense)                (None, 200)          40200       ['add_1[0][0]']

 dense_5 (Dense)                (None, 2009)         403809      ['dense_4[0][0]']

==================================================================================================
Total params: 2,308,945
Trainable params: 1,907,145
Non-trainable params: 401,800
__________________________________________________________________________________________________
```

6.3.2 Model Scores

We incorporate multi-head attention, which modifies the baseline model and provides a significant boost to both the BLEU Score and Meteor Score.

	Averaged Score	Maximal Score
BLEU Score	0.5990	1.0
METEOR Score	0.3824	0.9563

6.3.3 Plot Modified Baselined Model


input_5 InputLayer | input: [(None, 38)] | output: [(None, 38)]
embedding_1 Embedding | input: (None, 38) | output: (None, 38, 200)
lstm_1 LSTM | input: (None, 38, 200) | output: (None, 38, 200)
multi_head_attention MultiHeadAttention | input: (None, 38, 200) | output: (None, 38, 200)
dropout_3 Dropout | input: (None, 38, 200) | output: (None, 38, 200)
layer_normalization LayerNormalization | input: (None, 38, 200) | output: (None, 38, 200)
input_4 InputLayer | input: [(None, 2048)] | output: [(None, 2048)]
dropout_2 Dropout | input: (None, 2048) | output: (None, 2048)
dense_3 Dense | input: (None, 2048) | output: (None, 200)
reshape_1 Reshape | input: (None, 200) | output: (None, 1, 200)
concatenate_1 Concatenate | input: [(None, 38, 200), (None, 1, 200)] | output: (None, 39, 200)
lstm_2 LSTM | input: (None, 39, 200) | output: (None, 200)
dropout_4 Dropout | input: (None, 200) | output: (None, 200)
add_1 Add | input: [(None, 200), (None, 200)] | output: (None, 200)
dense_4 Dense | input: (None, 200) | output: (None, 200)
dense_5 Dense | input: (None, 200) | output: (None, 2009)

6.3.4 Predictions Using Modified Baselined Model

Prediction-1: a black dog be run through the snow.

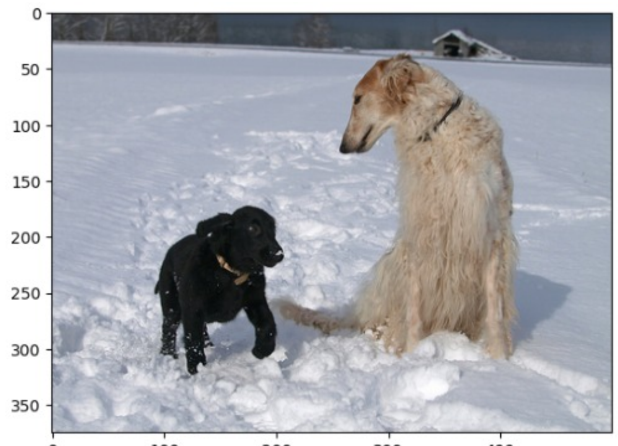
Training Sample-1 startseq a big white dog and a small black dog sit in the snow endseq

Training Sample-2 startseq a large white dog watch a little black dog run in the snow endseq.

Training Sample-3 startseq a small dog and a large dog in the snow endseq.

Training Sample-4 startseq a tall white dog sit and watch a small black dog run around him in the snow endseq.

Training Sample-5 startseq a white dog look at a black puppy in the snow endseq



a black dog be run through the snow

Prediction-2: a basketball player in a white uniform be block a basketball in a basketball game

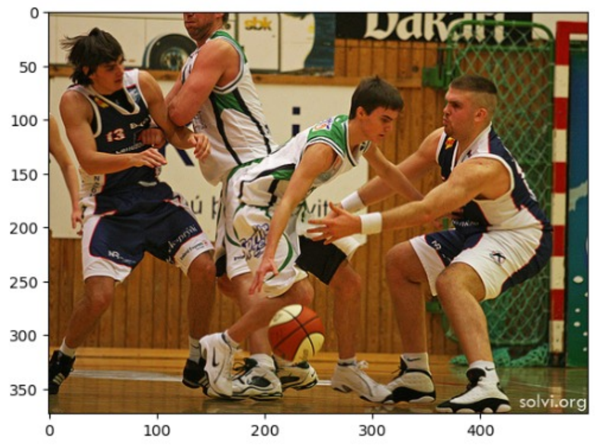
Training Sample-1 startseq a player from white and green highschool team dribble down court defend by a player from the other team endseq.

Training Sample-2 startseq four basketball player in action endseq.

Training Sample-3 startseq four man play basketball two from each team endseq

Training Sample-4 startseq two boy in green and white uniform play basketball with two boy in blue and white uniform endseq

Training Sample-5 startseq young man play basketball in a competition endseq



a basketball player in a white uniform be block a basketball in a basketball game

7 Comparison between Baselined and Modified Baselined (Analysis)

7.1 Differences observed with the applied technique.

We incorporate multi-head attention, which modifies the baseline model and provides a significant boost to both the BLEU Score and Meteor Score.

The Scores of Baselined can be found 6.2.2 and the Scores of Modified Baselined can be found 6.3.2.

Upon examining the generated captions produced by both Baselined and Modified Baselined models, it is evident from a visual standpoint that the Modified Baselined model exhibits a greater capacity to extract a richer array of features, thus enabling a more comprehensive exploration of the entire image. This observation substantiates the enhanced performance and heightened analytical capabilities of the Modified Baselined model. The predictions of baselined can be found 6.2.4 and predications of modified baselined can be found 6.3.4.

First lets know about attention and the multi-head attention Layer:

- **Attention mechanism**: The attention mechanism allows the model to focus on different parts of the visual features while generating each word of the caption. This helps the model align the visual and textual information more effectively. By attending to different regions of the image, the model can associate relevant visual information with corresponding words in the caption.
- **Multihead attention**: The multi-head attention mechanism improves upon the standard attention mechanism by using multiple attention heads. Each attention head learns to attend to different subsets of visual features and generate corresponding context vectors. The outputs of the attention heads are then concatenated and combined to produce the final attended features. This multi-head approach allows the model to capture different types of visual information simultaneously and capture more complex relationships between the image and the generated caption.

7.2 Subjective Comparison

7.2.1 Why did the BLEU Score Increase in Modified Baselined?

- *Improved alignment:* The model's attention mechanism enables it to align the generated words with the necessary visual elements. The model may better correlate the visual information with the appropriate words in the caption by paying attention to different portions of the image. This enhanced alignment can result in captions that are more accurate and contextually appropriate, which can boost the BLEU score.
- *Enhanced context understanding:* The multi-head attention mechanism simultaneously gathers diverse types of visual information and intricate links between the image and the generated caption. Because of this improved context awareness, the model can generate captions that are more loyal to the visual content and more accurately capture the semantics of the image. As BLEU compares the generated caption to one or more reference captions, the model's context improves.
- *Increased diversity in generated captions:* The model may attend to diverse subsets of visual cues via the multi-head attention method, resulting in multiple context vectors from the different attention heads. This enhanced diversity in attended features has the potential to generate various subtitles. BLEU encourages diversity by taking into account the n-gram overlap between the generated and reference captions. The BLEU score can be greater if the model creates many diverse captions that contain distinct valid n-grams present in the reference captions.
- *Better handling of long sentences:* Even for long sentences, the attention mechanism can assist the model in focusing on essential visual elements while creating each word of the caption. This increased concentration may prevent the model from producing overly generic or unrelated terms, which BLEU may penalize. By paying attention to the important visual elements at each stage, the model may generate captions that are more cohesive and fluid, resulting in a higher BLEU score.

7.2.2 Why did the METEOR Score Increase in Modified Baselined?

- *Improved language understanding:* The multi-head mechanism's increased attentiveness can lead to better language comprehension. The algorithm can generate more accurate and contextually appropriate captions by attending to multiple visual regions and gathering diverse visual information. This can have a favorable impact on evaluation criteria like the Meteor score, which rates the quality of output captions by taking precision and recall into account.
- *Improved precision and recall:* Meteor score consider n-gram matches between the generated caption and the reference captions to account for both precision and recall. The multi-head attention mechanism assists the model in attending to relevant visual elements when producing captions, resulting in greater alignment between visual material and generated words. Improved alignment can lead to higher precision and recall levels, which can raise the Meteor score.
- *Handling synonymy and paraphrasing:* Meteor score uses WordNet-based matching and stemming from adding synonymy and paraphrasing. By gathering various visual inputs, the multi-head attention layer can give the model a fuller representation of the image content. This can assist the model in generating captions that express the same meaning using alternative words or phrases, which can help achieve higher Meteor scores.