Visual Recognition

Mini Project

Image Captioning CNN-LSTM

Team ID - 40 :

Ankrutee Arora [IMT2020034]

Ashish Gatreddi [IMT2020073]

Tejdeep Gutta [IMT2020102]

Overview

The goal of this project is to develop a robust CNN-LSTM model for image captioning using the Flickr8 dataset. The approach involves implementing a baseline model with a chosen CNN-LSTM architecture and evaluating its performance, we then go on to improve the image and word embeddings while keeping the LSTM architecture the same. The objective is to create a highly accurate and efficient image captioning system that can produce descriptive and natural-sounding captions for a wide range of images. The performance of the baseline as well as modified models are evaluated using various metrics, such as BLEU and METEOR scores.

Baseline Model

The baseline model for image captioning consists of a Convolutional Neural Network (CNN) for image feature extraction (vision embeddings) and a Long Short-Term Memory (LSTM) network for language modeling (word embeddings). The CNN extracts features from the input image, and the LSTM generates captions based on these features.

Enhanced Embeddings

To improve upon the baseline model, the vision embeddings and language embeddings are enhanced without making any architectural modifications to the LSTM. Vision embeddings are representations of the input image, while language embeddings are representations of the output captions. The enhanced embeddings are designed to capture more meaningful and relevant information, leading to better performance in image captioning tasks.

Evaluation Metrics

The effectiveness of the enhanced embeddings is compared to the original baseline using various evaluation metrics, such as BLEU and METEOR scores. These metrics are widely used in the evaluation of image captioning systems and provide insights into the quality of generated captions. BLEU measures the n-gram overlap between the generated caption and the ground truth caption, while METEOR measures the overall quality of the generated caption based on a set of pre-defined criteria. We discuss these scores in detail in a different section.

Flickr8K dataset

The Flickr8K dataset is a widely used benchmark dataset for image captioning. It consists of 8,000 images that were collected from six different Flickr groups and are accompanied by five different captions each. The dataset is designed to cover a wide range of scenarios and circumstances, providing a diverse set of images and captions to train and test image captioning models. One of the key strengths of the Flickr8K dataset is the fact that it does not feature any well-known persons or places, ensuring that models trained on this dataset are not biased towards specific entities or locations. For each image we have 5 captions provided by 5 different humans.

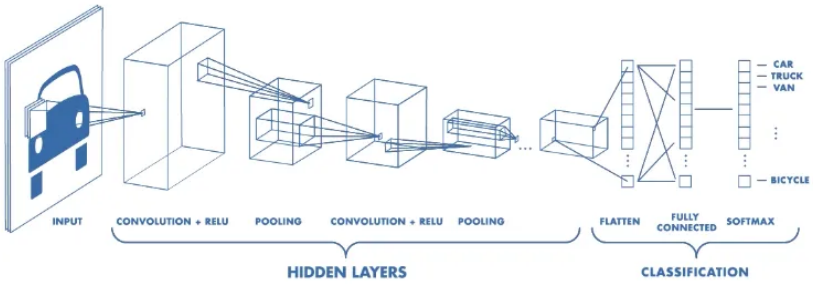
	**Caption1:** একটি কালো এবং সাদা পাখি একটি হাতে দাঁড়িয়ে। (A black and white bird stands with one hand.) **Caption2:** একটি কালো এবং সাদা পাখি কারও হাত থেকে বীজ খাচ্ছে (A black and white bird is eating seeds from someone's hand) **Caption3:** একটি ছোট পাখি কোনও ব্যক্তির হাতে বসে বীজ খায়। (A small bird sits in a person's hand and eats seeds.) **Caption4:** কালো ও সাদা পাখি কারও হাতে দাঁড়িয়ে আছে সূর্যমুখী বীজ। (Black and white birds are standing in someone's hand with sunflower seeds.) **Caption5:** ছোট পাখি হাতের আঙুলের সাথে বীজ ধরে আছে। (Small birds are holding seeds with their fingers.)
	**Caption1:** তিন যুবতী লনে বসে আছেন। (Three young women are sitting on the lawn.) **Caption2:** তিনটি ছোট বাচ্চা ঘাসে বসে আছে। (Three small children are sitting on the grass.) **Caption3:** তিনটি ছোট মেয়ে ঘাসে বসে আছে। (Three little girls are sitting on the grass.) **Caption4:** তিনটি শিশু ঘাসের উপর বসে । (Three children sit on the grass.) **Caption5:** তিনটি মেয়ে ঝোপের সামনে ঘাসের উপরে বসে আছে। (The three girls are sitting on the grass in front of the bushes.)
	**Caption1:** একটি কালো কুকুর এবং একটি কালো কুকুর একটি সকার বলের উপর লড়াই করছে । (A black dog and a black dog are fighting over a soccer ball.) **Caption2:** দুটি কুকুর একটি নীল বলের উপরে খেলে এবং লড়াই করে । (Two dogs play and fight over a blue ball.) **Caption3:** দুটি কুকুর নীল এবং সবুজ বল খেলছে । (Two dogs playing blue and green balls.) **Caption4:** দুটি কুকুর নীল এবং হলুদ রঙের বলটিতে কুঁচকে যাচ্ছে । (Two dogs are going to wrinkle in the blue and yellow ball.) **Caption5:** দুটি কুকুর বল নিয়ে খেলছে । (Two dogs playing with the ball.)

CNN :

A Convolutional Neural Network (CNN), also known as ConvNet, is a specialized type of neural network architecture mainly used for processing and analyzing image data. When we refer to a digital image, we mean a representation of visual information using binary data. It consists of a grid-like structure composed of individual picture elements, or pixels, which store values representing brightness and color.

CNNs employ various layers to perform different operations. The fundamental operations in a CNN include convolutional layers, pooling layers, and fully connected layers. Convolutional layers apply filters or kernels to input images, extracting features through convolution operations. Pooling layers reduce the spatial dimensions of feature maps, focusing on the most relevant information. Fully connected layers establish connections between the extracted features and the final classification or regression layers.

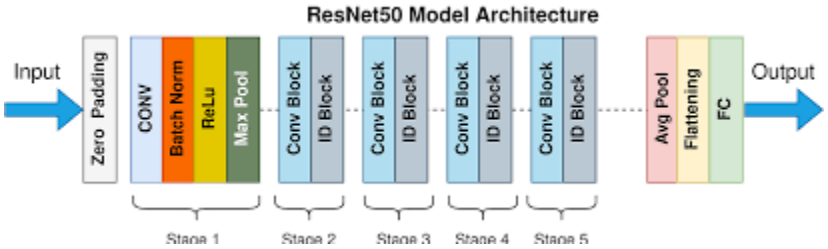
By stacking multiple convolutional and pooling layers, CNNs progressively capture more complex patterns and features present in images. This hierarchical learning enables them to learn essential visual representations and make predictions based on those representations.



These are the cnn architectures we will be using in our project.

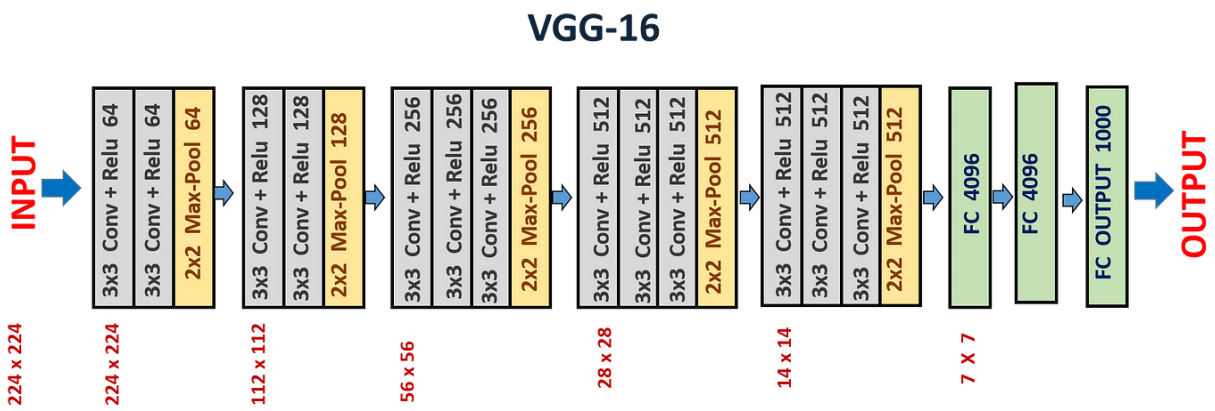
ResNet50 Architecture

ResNet50 specifically refers to a ResNet model with 50 layers. Instead of relying solely on stacked convolutional layers, ResNet introduces residual blocks. These blocks contain skip connections that allow information to bypass one or more layers, creating shortcut paths for the flow of gradients during training. The main benefit of residual connections is that they alleviate the vanishing gradient problem that can occur in deep networks. By allowing the gradients to flow more easily, ResNet50 can be trained effectively even with its significant depth. ResNet50 utilizes 3x3 convolutional kernels and occasional 1x1 convolutions for dimensionality reduction. The network also includes max-pooling layers for downsampling. Like VGGNet, pre-trained versions of ResNet50 are available, often trained on large datasets such as ImageNet. These pre-trained models capture rich and generalizable features from images. For our purposes, we can utilize ResNet50 for feature extraction by removing the final classification layer. This will provide us with a feature vector of 2048 dimensions for each image, which can be used for various downstream tasks without the need for retraining the entire network.



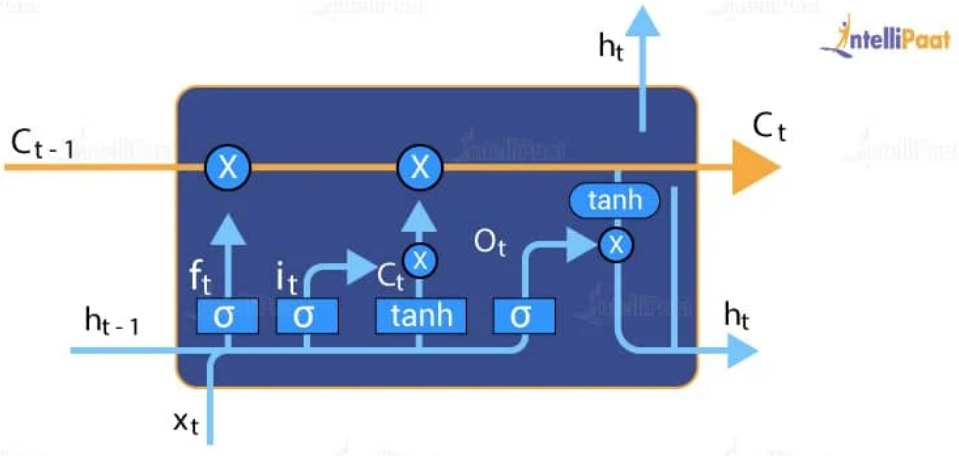
VGG16 Architecture:

The approach was developed with the aim of reducing parameters in convolutional layers and minimizing training time. VGGNet, which includes variations such as VGG16 and VGG19, differs solely in the total number of layers. VGG16, for instance, has 138 million parameters. The network utilizes 3x3 convolutional kernels and 2x2 max-pooling kernels with a stride of two. To obtain meaningful features from images, it is advantageous to employ pre-trained deep convolutional neural networks (CNNs) like Tensorflow's VGG-16. These networks are typically trained on extensive datasets such as ImageNet and are capable of generating excellent features. In our case, we are interested in the second-to-last layer of the model, which yields a 4096-dimensional feature vector for each image. The final layer, designed for classification purposes with 1000 units corresponding to the ImageNet dataset, is not required for our specific task.



LSTM:

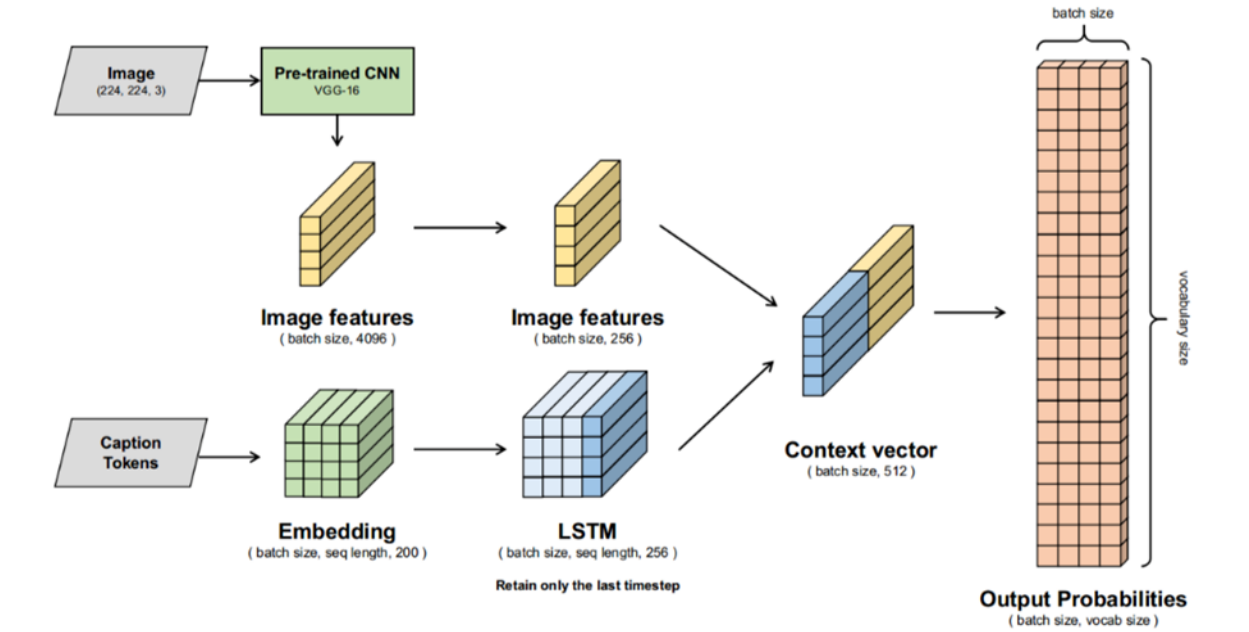
Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture that is particularly effective in capturing dependencies in sequential data. It was designed to address the limitations of traditional RNNs, which struggle with long-term dependencies. At the core of an LSTM is the concept of a memory cell. The memory cell is responsible for storing and updating information over time, allowing the network to retain important information and ignore irrelevant details. The memory cell has the ability to add or remove information through a process called "forgetting" and "adding." This way, it can selectively remember or forget information as it processes the sequence. LSTMs also have a hidden state, which acts as the network's memory. The hidden state carries information from previous time steps and influences the computations at the current time step. By using memory cells and hidden states, LSTMs can effectively capture and propagate information through sequential data, enabling them to model long-term dependencies.



CNN+LSTM model:

The CNN-LSTM model is a hybrid architecture that combines the power of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks.

- It is used for tasks like generating captions for images or videos. When working with images, CNNs are excellent at extracting important features by applying convolutional filters. They can capture local patterns and build hierarchical representations, making them popular for tasks such as image classification and object detection.
- LSTM networks are designed for sequence prediction and struggle to directly process spatial data like images.
- To overcome this limitation, the CNN-LSTM model integrates CNNs into the architecture.



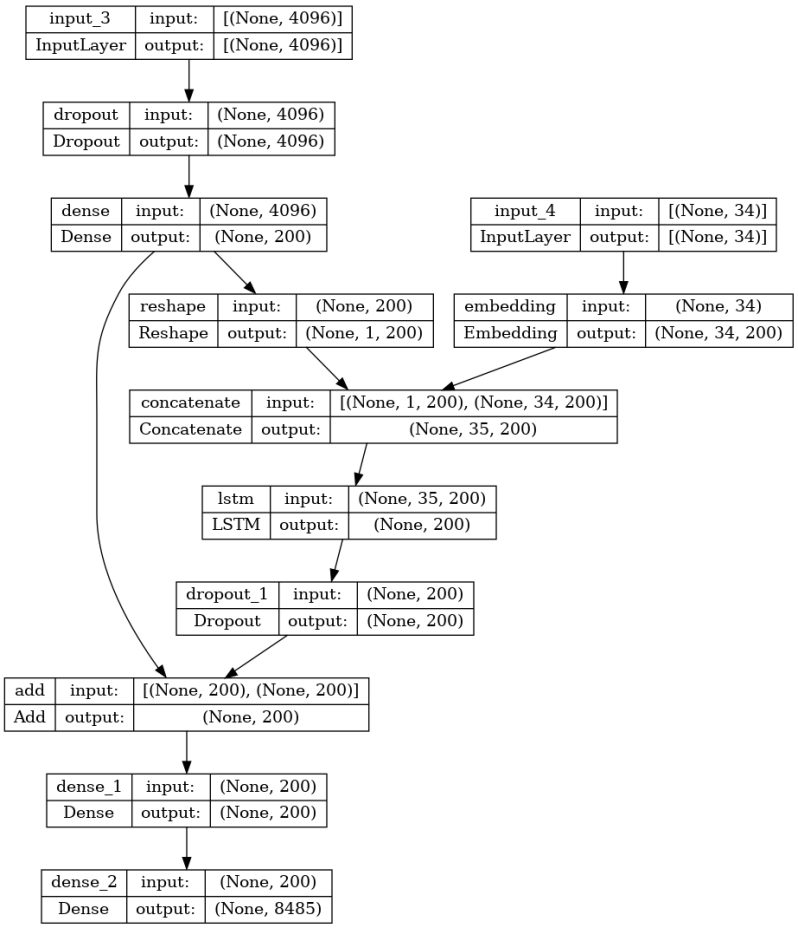
General Structure:

1. CNNs are employed to extract relevant features from the input images, which are then fed into LSTM layers for sequence prediction.
2. The convolutional layers apply filters, extracting spatial features and producing a feature map. This feature map contains higher-level representations of the input image.
3. The feature map is then passed on to the LSTM layers, which are capable of capturing and modeling temporal dependencies in the sequence.

4. The LSTM layers take the feature vectors as input and generate the desired sequence predictions, such as generating captions for images.
5. By combining the strengths of CNNs in extracting spatial features and LSTMs in modeling sequential dependencies, the CNN-LSTM model can generate accurate and meaningful predictions for tasks that involve both visual and sequential information.

Baseline Model

We have used VGG16 as the convolutional network for our baseline architecture, embeddings are made to be of size 256 and the word vocabulary after preprocessing is 8485.



Baseline:

```
Model: "model_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_9 (InputLayer)        [(None, 2048)]            0

 reshape_3 (Reshape)         (None, 1, 2048)           0

 dropout_7 (Dropout)         (None, 1, 2048)           0

 input_10 (InputLayer)       [(None, 36)]              0

 dense_9 (Dense)             (None, 1, 200)            409800

 embedding_3 (Embedding)     (None, 36, 200)           1455400

 reshape_4 (Reshape)         (None, 1, 200)            0

 dense_10 (Dense)            (None, 36, 200)           40200

 concatenate_3 (Concatenate)  (None, 37, 200)          0

 dense_11 (Dense)            (None, 37, 200)           40200

 dense_12 (Dense)            (None, 37, 200)           40200

 dense_13 (Dense)            (None, 37, 200)           40200

 attention (Attention)       (None, 37, 200)           0

 dropout_8 (Dropout)         (None, 37, 200)           0

 lstm_5 (LSTM)               (None, 37, 200)           320800

 dropout_9 (Dropout)         (None, 37, 200)           0

 lstm_6 (LSTM)               (None, 37, 200)           320800

 dropout_10 (Dropout)        (None, 37, 200)           0

 reshape_5 (Reshape)         (None, 200)               0

 lstm_7 (LSTM)               (None, 200)               320800

 add_3 (Add)                 (None, 200)               0

 dense_14 (Dense)            (None, 200)               40200

 dense_15 (Dense)            (None, 7277)              1462677

=================================================================
Total params: 4,491,277
Trainable params: 4,491,277
Non-trainable params: 0
_________________________________________________________________
```

Attention:

```
Model: "model_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_26 (InputLayer)       [(None, 36)]              0

 embedding_8 (Embedding)     (None, 36, 200)           1455400

 input_25 (InputLayer)       [(None, 2048)]            0

 lstm_16 (LSTM)              (None, 36, 200)           320800

 dropout_22 (Dropout)        (None, 2048)              0

 multi_head_attention_4 (MultiH  (None, 36, 200)       411336
 eadAttention)

 dense_24 (Dense)            (None, 200)               409800

 dropout_23 (Dropout)        (None, 36, 200)           0

 layer_normalization_2 (LayerNo  (None, 36, 200)       400
 rmalization)

 reshape_10 (Reshape)        (None, 1, 200)            0

 concatenate_8 (Concatenate)  (None, 37, 200)          0

 lstm_17 (LSTM)              (None, 200)               320800

 dropout_24 (Dropout)        (None, 200)               0

 add_6 (Add)                 (None, 200)               0

 dense_25 (Dense)            (None, 200)               40200

 dense_26 (Dense)            (None, 7277)              1462677

=================================================================
Total params: 4,421,413
Trainable params: 4,421,413
Non-trainable params: 0
```

Preprocessing images -

We use basic preprocessing, since VGG16 takes inputs of size (224,224) we reshape our inputs to these dimensions and then collect the features.

Preprocessing captions -

First we read the “Flickr8k.token.txt” file using readlines which gives us an array that constraints elements which are separated by newline in the text file. Now, each line has the image name, and caption number and the caption, so we split and get the image names and captions respectively. Once we have these, we map each image name (ID) to an array of captions that correspond to that image.

Once the mapping is done, we clean the mappings - Essentially we convert the sentence case to lowercase and add start and end tokens (“startseq” and “endseq”) that denote the beginning and the ending of the sentence respectively. We then remove any other symbols other than alphabets from the sentence as well as the extra spaces.

Baseline scores -

We have used **corpus_bleu** as the metric for evaluation -

	BLEU - 1	BLEU - 2	BLEU - 3	BLEU - 4	METEOR
VGG16	0.514331	0.302212	0.183430	0.128440	0.27003
ResNet50	0.532124	0.312067	0.189912	0.12969	0.29897

Now, we discuss the various techniques we have used to enhance the embeddings -

First we tried using glove embeddings to initialize weights for calculating the word embeddings for captions.There are several advantages of using GloVe (Global Vectors for Word Representation) in image captioning:

Better representation of semantic relationships: GloVe is based on a co-occurrence matrix that captures the semantic relationships between words in a corpus. By using GloVe in image captioning, the model can better understand the semantic relationships between words in the caption and the visual features of the image.

Better performance on out-of-vocabulary words: GloVe embeddings can be pre-trained on large text corpora, which makes them effective in handling out-of-vocabulary words. This is particularly important in image captioning, where the model may encounter new or rare words that were not seen during training.

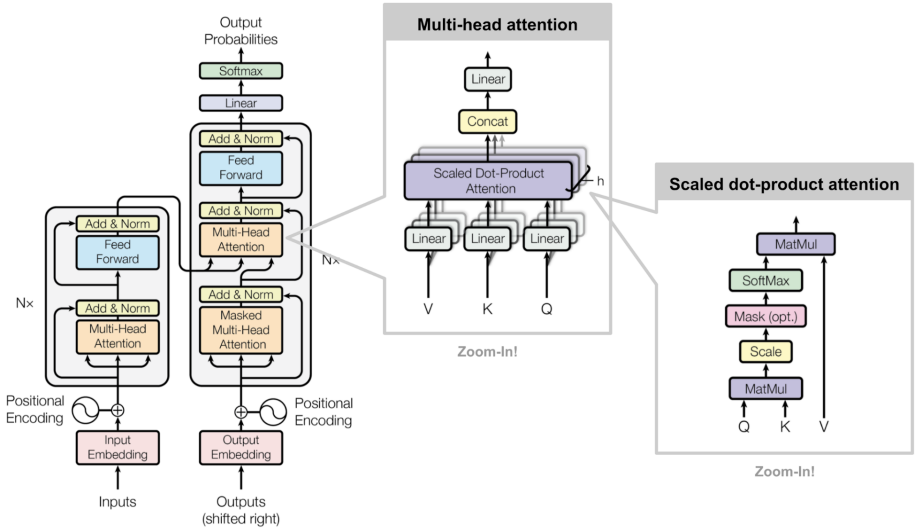
Faster training and better convergence: GloVe embeddings can be used to initialize the word embeddings of the decoder in a neural network model, which can help with faster training and better convergence. This is because the initialized embeddings are already semantically meaningful and can better capture the relationships between words in the caption and the visual features of the image.

Ability to incorporate multiple modalities: GloVe can be combined with other types of embeddings, such as visual embeddings, to create multimodal embeddings that represent both the visual and textual aspects of an image. These embeddings can be used in a neural network model to generate more accurate and descriptive captions.

Overall, the advantages of using GloVe in image captioning include better representation of semantic relationships, better performance on out-of-vocabulary words, faster training and better convergence, and the ability to incorporate multiple modalities.

With Glove we didn't see a great change in the output scores but observed that the convergence to get optimal output has become faster, so running a lesser number of epochs is sufficient.

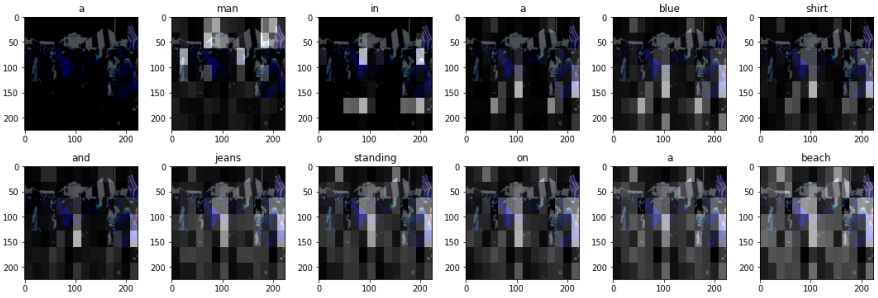
1) Attention :



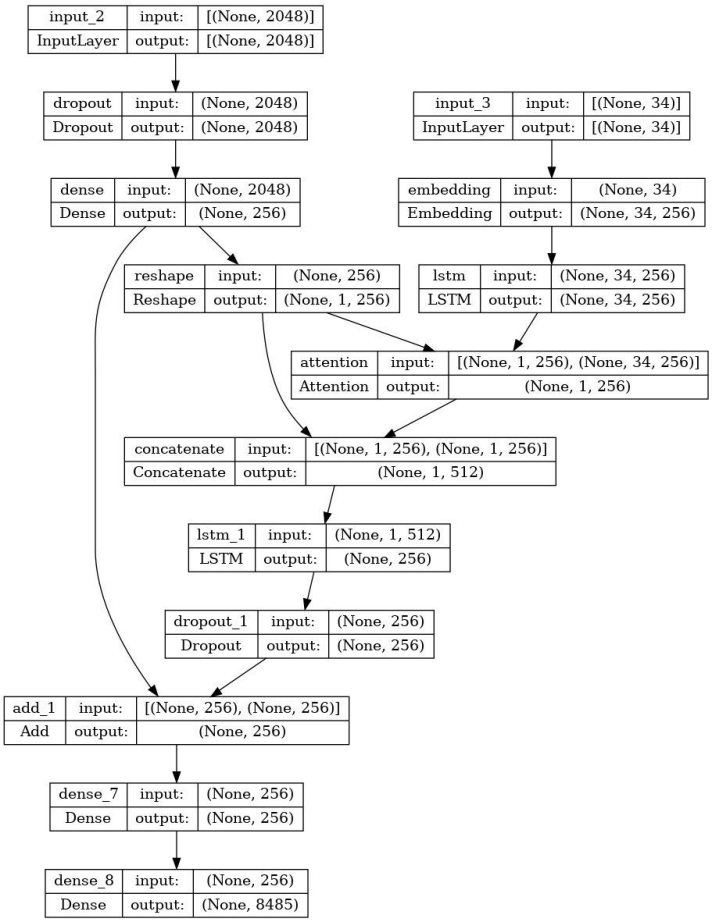
The attention mechanism enables the model to focus on specific parts of the input sequence while making predictions or generating output, improving the model's ability to capture relevant information. In traditional sequence-to-sequence models, such as encoder-decoder architectures, a fixed-length representation of the entire input sequence is encoded into a fixed-length vector. However, this approach may be suboptimal when dealing with long sequences or when certain parts of the sequence are more important than others.

The attention mechanism addresses this limitation by allowing the model to dynamically pay attention to different parts of the input sequence when making predictions. It achieves this by assigning different weights, or attention scores, to each element in the input sequence, indicating their importance or relevance.

When we use attention on the CNN output with LSTM, we essentially are calculating self attention for the concatenated vision and word embeddings(input embedding). Hence we are focusing on different aspects of the image while predicting a certain word.



Architecture -



Here we have used this architecture for both VGG16 and ResNet50 as the CNNs.

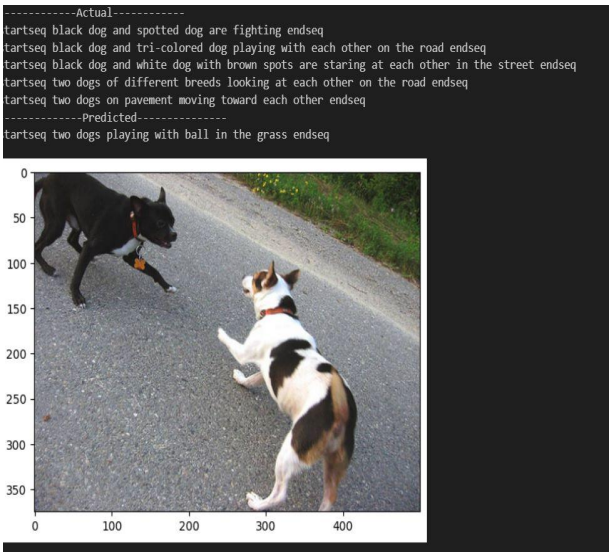
The corresponding corpus_bleu scores are -

	BLEU - 1	BLEU - 2	BLEU - 3	BLEU - 4	METEOR
ResNet50	0.56778	0.34546	0.24645	0.16432	0.312564
VGG16	0.54332	0.33298	0.22998	0.14143	0.304781

Why the attention model performs better :

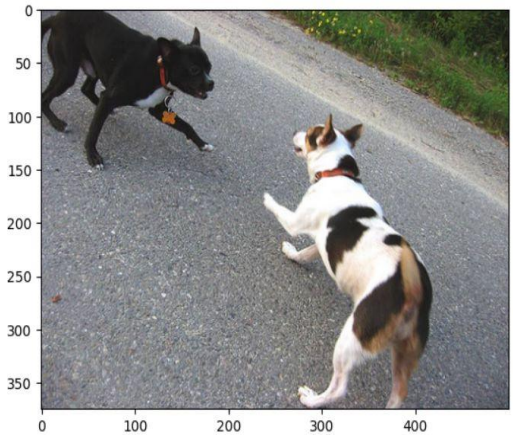
The decoder dynamically focuses on various areas of the image while each word of the caption is being generated. Attention weights are computed using the self-attention query,key,value equations, these weights indicate the importance of the region. Attention weights are tuned during training. Using the attention weights, the attention mechanism creates a weighted sum of the image features. This weighted sum is used in calculating the **context** part of the LSTM block, the context part is calculated by using the image vector as well as the weights, the context vector is then combined with the hidden state to make predictions for the next word. The attention mechanism helps **handle varying content** and **long captions** and also provide a form of **localisation.**

1. **Baseline**

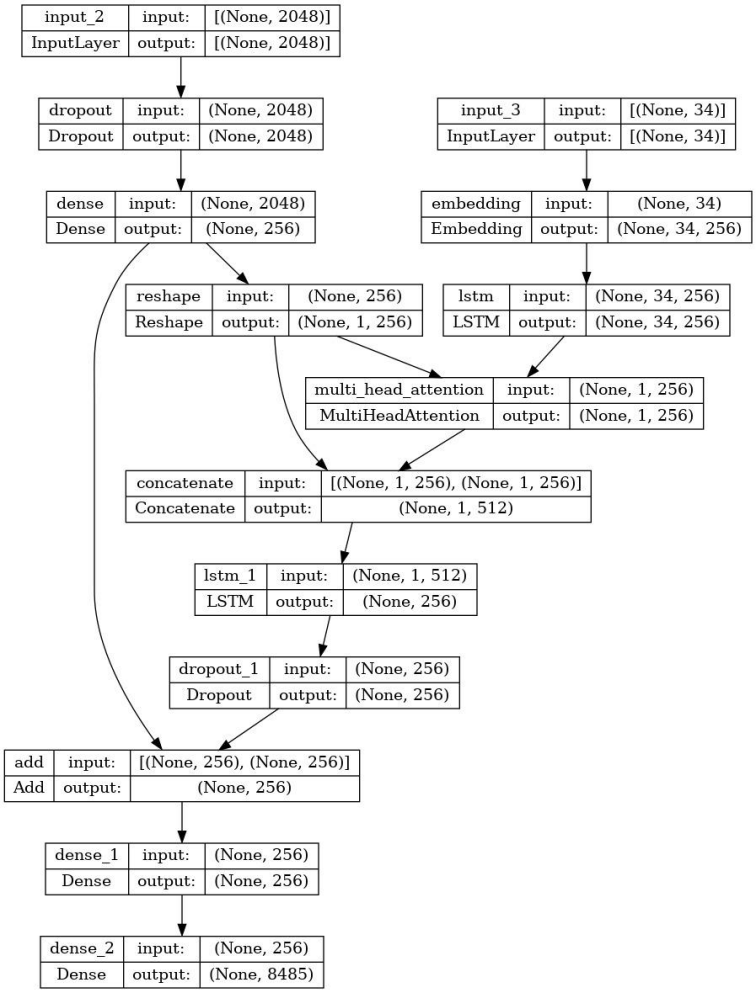


2. **Attention**

```
------------Actual------------
startseq black dog and spotted dog are fighting endseq
startseq black dog and tri-colored dog playing with each other on the road endseq
startseq black dog and white dog with brown spots are staring at each other in the street endseq
startseq two dogs of different breeds looking at each other on the road endseq
startseq two dogs on pavement moving toward each other endseq
-------------Predicted---------------
startseq two dogs play together in the grass endseq
```



2) Multi-head attention -



	BLEU - 1	BLEU - 2	BLEU - 3	BLEU - 4	METEOR
ResNet50	0.57679	0.35241	0.25531	0.16932	0.32214
VGG16	0.54526	0.34454	0.23005	0.15123	0.292345

Enhanced representation learning: Multi-head attention allows the model to focus on different aspects or patterns of the input sequence simultaneously. This can help capture more diverse and informative features, leading to better representation learning.

Increased modeling capacity: By employing multiple attention heads, the model has a higher capacity to capture complex relationships and dependencies within the input sequence. Each attention head can attend to different parts of the sequence, allowing the model to extract more fine-grained information.

Improved generalization: The ability of multi-head attention to attend to different parts of the sequence and capture diverse features can enhance the model's generalization capability. It enables the model to effectively learn meaningful representations from the input, which can generalize well to unseen data.

BLEU and METEOR

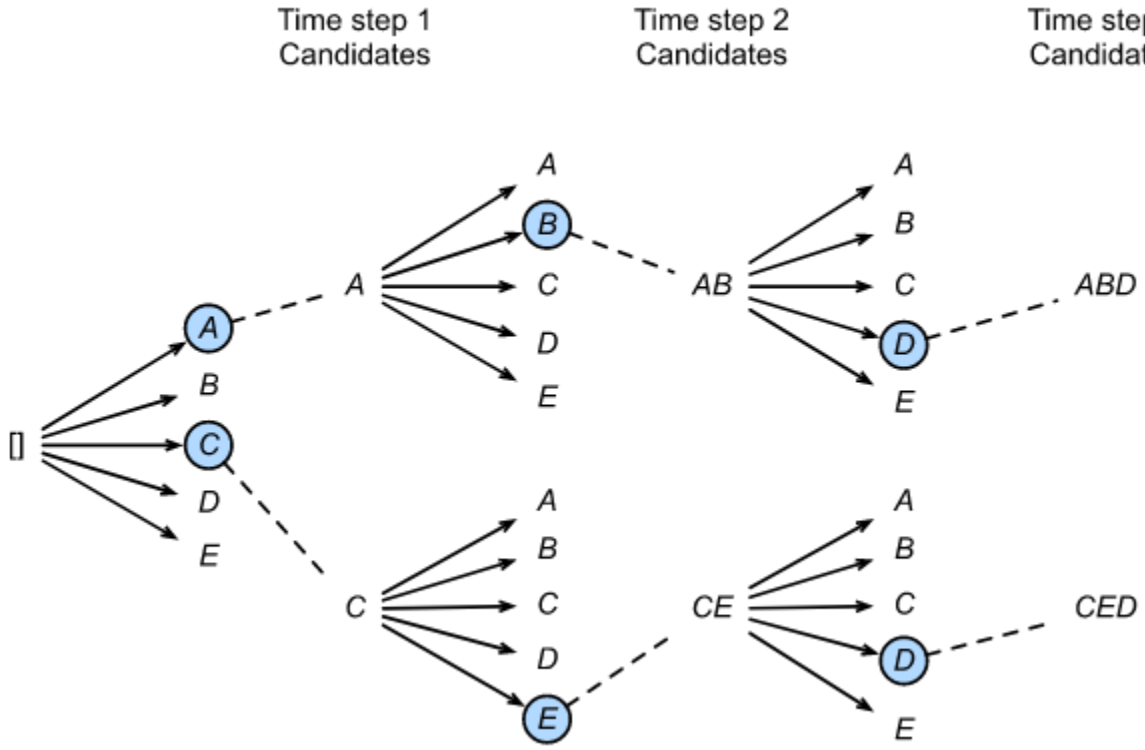
BLEU (Bilingual Evaluation Understudy) and METEOR (Metric for Evaluation of Translation with Explicit ORdering) are commonly used evaluation metrics for assessing the quality of machine-generated text, such as image captions. These metrics compare the generated captions with reference captions (ground truth) and provide objective scores to measure the similarity and quality of the generated captions.

BLEU Score: BLEU is a precision-based metric that compares the n-gram overlap between the generated and reference captions. It calculates the precision of n-grams (typically 1 to 4) in the generated caption by comparing them with the reference captions. The BLEU score ranges from 0 to 1, with a higher score indicating better similarity between the generated and reference captions.

METEOR Score: METEOR is a metric that considers both precision and recall by incorporating additional linguistic features. It measures the alignment between the generated and reference captions, taking into account word order, stemming, synonyms, and other linguistic variations. The METEOR score also ranges from 0 to 1, with a higher score indicating better similarity between the generated and reference captions.

Beam Search:

Beam search is a heuristic algorithm used in natural language processing and machine translation tasks, particularly in sequence generation tasks such as text generation or language translation. It is used to find the most likely output sequence given a probabilistic model. In beam search, instead of exhaustively exploring all possible output sequences, the algorithm maintains a set of the most promising candidates at each step of generation. These candidates are referred to as the "beam." The beam width or size determines the number of candidates that are kept at each step.



Basic Algorithm

1. The algorithm starts with an initial input and generates all possible next steps based on the current candidates.
2. Each candidate is assigned a score based on the model's probability distribution.
3. The candidates with the highest scores are selected to be the new set of candidates for the next step, while the rest are pruned.
4. This process continues iteratively until a predefined stopping condition is met, such as generating a fixed number of output tokens or reaching an end-of-sequence token.

By limiting the number of candidates at each step, beam search balances exploration and exploitation. It explores different possible paths while still favoring the most likely ones based on the model's probabilities. The beam width parameter determines the trade-off between exploration and exploitation. A larger beam width allows for more exploration but also increases the computational complexity. Although we could not implement beam search fully due to lack of time, we do consider it as a method to improve upon the baseline.

The problem of Language Bias:

The issue of our model consistently predicting the word "dog" for many images can be attributed to a phenomenon called "mode collapse." Mode collapse occurs when the model repeatedly outputs the same word regardless of the input image. This can happen due to several factors, including insufficient training data, limitations in the model architecture, or a lack of diversity in the training examples.

One possible reason for the overprediction of the word "dog" could be an imbalance in the dataset, with a disproportionately high number of captions containing the word "dog." As a result, the model may perceive "dog" as a common and likely label, leading to its repetitive prediction.

Ex:

Real captions:

A black dog on a rocky beach.

A black dog walks along an ocean front.

A black dog walks on the beach near the rocks.

A black dog walks on the rocks.

Black dog walking on the beach after swimming in the ocean.

Predicted caption: A black dog running on the sand in the beach.

Nouns in real caption: black dog, rocks, beach, swimming in the ocean, ocean front, rocky beach.

Nouns in the predicted caption: black dog, beach, sand.

Observation: out of the 3 nouns in the predicted caption, 2 were present in the real captions too. The LSTM model was trained over a lot of images containing sand so it got biased. Although there is no presence of sand in the captions of the image, the predicted caption contains the noun water.

To address this problem, several strategies can be employed:

1. Increasing training data: Expanding the dataset used for training can provide the model with a wider range of examples, allowing it to learn more diverse image-caption relationships. This can help mitigate the bias towards predicting "dog" inappropriately.

2. Architecture modifications: Exploring more advanced models, such as Transformer-based architectures, may improve the model's ability to generate diverse and contextually relevant captions. These models are designed to capture complex dependencies and long-range contextual information, potentially mitigating mode collapse issues.

3. Data augmentation: Applying techniques such as image transformations, cropping, or adding noise to the training data can introduce additional variations. By augmenting the dataset, the model can be exposed to a greater variety of images and corresponding captions, reducing the likelihood of mode collapse and encouraging more diverse predictions.

4. Regularization techniques: Applying regularization techniques during training, such as dropout, batch normalization, or adding noise to the model's parameters, can help prevent overfitting and encourage the model to explore different captioning options. Regularization can promote diversity in predictions and reduce the dominance of specific words like "dog."

It is worth noting that in cases where the model is trained with label encoding, the repetition of certain words may also arise from the high frequency of those words in the dictionary. In such instances,

techniques like data balancing or adjusting the label encoding scheme can be considered to address the issue.

Instructions to run the code

Requirements:-

1. Keras
2. Tensorflow
3. Numpy
4. Pandas
5. tqdm
6. nltk
7. Pickle

Running the code:-

1. In order to run the baseline model, use baseline.ipynb, we have used kaggle to run our code and saved the flicker dataset file as “flickr8vr1” and passed it onto “kaggle/input” in order to run the code appropriate adjustments might be needed as per the dataset name that is being used.
2. To run the attention model, open modifiedb-attention.ipynb and run all the cells. Here also we have used “flickr8vr1” . Testing and training are done in the same file, the attention architectures are commented out, the user can uncomment and run whichever architecture they choose to.
3. To run the glove model, open modified-glove.ipynb and rest is similar to attention notebook.

Further improving the model:

1. Utilize advanced word embedding techniques: Instead of relying solely on label encoding, incorporate techniques like Word2Vec or BERT to generate word embeddings. These methods capture the semantic relationships between words and can improve the model's understanding of captions, leading to more contextually accurate predictions.

2. Extend training duration: Consider training the model for a longer period, allowing it to learn more from the data and refine its predictions. However, be cautious of overfitting and find the optimal balance between training time and model performance.

3. Explore alternative visual feature extraction models: Experiment with different neural network architectures like VGGNet or Inception net, alongside ResNet, to compare their performance. Assess factors such as training time and evaluation metrics to determine which architecture best suits the task at hand.

4. Fine-tune pretrained GloVe embeddings: If using the GloVe embedding model, go beyond simply using the pretrained weights in nn.Embedding. Treat these weights as trainable parameters by setting requires_grad to true. This enables them to be fine-tuned during backpropagation, allowing the embeddings to adapt better to the specific dataset and improve model performance.

Here we look at two images (first - Glove, second - Baseline), we can see that the model does better with GloVe embeddings.

startseq little girl in red shirt is sitting in front of rainbow wheel endseq

startseq little girl is sitting in front of rainbow endseq

	BLEU - 1	BLEU - 2	BLEU - 3	BLEU - 4	METEOR
ResNet50	0.514331	0.302212	0.183430	0.128440	0.291691
VGG16	0.532124	0.312067	0.189912	0.12969	0.270673

References:

- https://www.youtube.com/watch?v=JTXPrjvhLl8
- https://www.tensorflow.org/api_docs/python/tf/keras/layers/MultiHeadAttention
- https://stackoverflow.com/questions/44324681/variation-in-bleu-score
- https://towardsdatascience.com/building-an-image-captioning-model-with-keras-ebccaadb98b9
- https://medium.com/@raman.shinde15/image-captioning-with-flickr8k-dataset-bleu-4bcba0b52926
- https://rpubs.com/vedpiyush/image-captioning-with-visual-attention
- https://www.analyticsvidhya.com/blog/2020/11/attention-mechanism-for-caption-generation/
- https://nlp.stanford.edu/projects/glove/