VISUAL RECOGNITION

VR PART-2 MINI PROJECT REPORT

IMT2020039 Anshul Jindal
IMT2020535 Shreeya Venneti
IMT2020094 Riddhi Chatterjee
IMT2020523 Kedar Deshpande

INTRODUCTION

Our goal is to develop a robust CNN-LSTM model for image captioning using the Flickr8 dataset. We will start by implementing a baseline model with a chosen CNN-LSTM architecture and evaluating its performance. To improve upon the baseline model, we will enhance both the vision embeddings and language embeddings without making any architectural modifications to the LSTM.

This modified baseline will then be compared to the original baseline using various evaluation metrics, such as BLEU and METEOR scores, to determine the effectiveness of the enhancements. Our aim 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 CNN-LSTM SYSTEM

CNNs

CNNs, or Convolutional Neural Networks, are a type of neural network primarily used for image and vision-based problems. A typical CNN comprises several layers, including the input layer, convolutional layer, pooling layer, activation layer, and fully connected layer.

The convolutional layer in a CNN applies filters or kernels to an input image, sliding them over the image with a stride to extract features. These filters are learned over the course of

training to produce better image representations. Pooling layers are used in CNNs to reduce the dimensionality of the layers, making subsequent computations more efficient. The activation layer introduces non-linearity in the computations using activation functions like sigmoid, RELU, ELU, etc.

The fully connected layer is the penultimate layer in a CNN and performs the final classification/regression task by computing a weighted sum of inputs and passing the result to the output layer. This layer contains the most dense weights in the network.

Thus, CNNs are highly effective at recognizing spatial patterns and hierarchical representations in images, making them useful for tasks such as image classification and feature extraction.

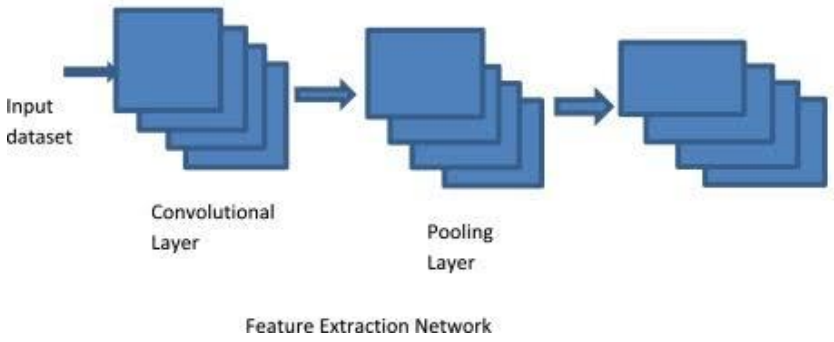


Figure 1. CNN as a feature extractor - the final blue layers are the feature maps

LSTMs

LSTM, or Long Short-Term Memory, is a type of neural network architecture that was developed as a modification of the RNN architecture. One of the main benefits of LSTM is its ability to overcome the vanishing gradient problem that is present in traditional RNNs. LSTMs excel in processing sequential inputs and can be applied to various tasks, including one-to-many, many-to-one, and many-to-many problems. They have proven useful in language-based applications, such as predicting the next word in a sentence, language translation, and combining vision and language, such as in image captioning when used with a CNN feature extractor. In case of language modeling task, an LSTM takes word embeddings as input and processes it sequence by sequence to learn the dependencies and patterns in the entire sequential input.

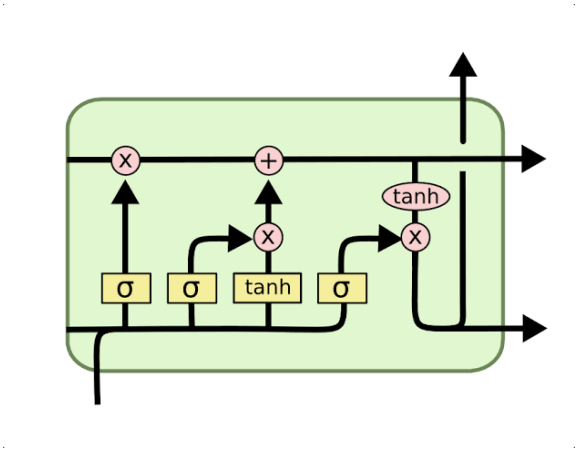


Figure 2. A typical LSTM

CNN + LSTM For Vision+Language

CNN + LSTM is used to merge visual feature extraction and sequential modeling for tasks involving vision and language. The visual features and corresponding text are encoded independently. The outputs are combined, creating a joint representation. A decoding layer generates the desired output, such as captions. This approach is applicable in image captioning, visual question answering, and similar applications.

CNN + LSTM For Image Captioning

Specifically when we come to the task of image captioning, the following strategy is used:

- CNN is first used for visual feature extraction. The multiple convolutional layers help capture and comprehend the visual patterns of the image in a hierarchical fashion
- The extracted visual features along with the encoded captions are passed as a sequence of inputs to the LSTM. Thus sequence encoding is implemented.
- In this way both modalities are fused, and the LSTM predicts the caption encodings keeping in mind the context of the image.
- The LSTM output is passed through a decoding layer which produces the predicted caption tokens.
- The weights of this entire network are trained and optimized to minimize a loss function (for eg : cross-entropy loss) via backpropagation

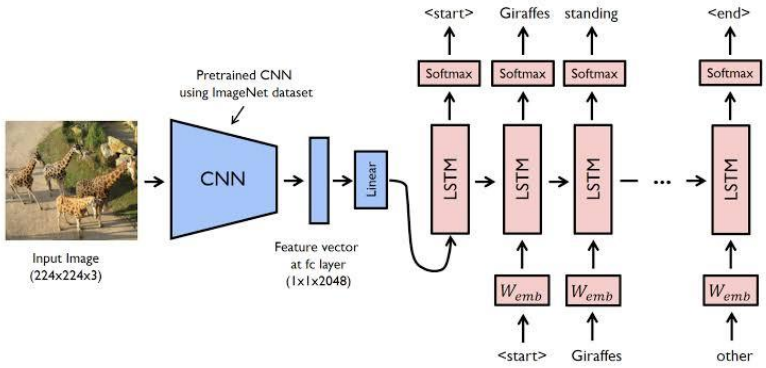


Figure 3. A CNN+LSTM image captioning network

1. BASELINE MODEL

PREPROCESSING

Setting up the captions

Every image from the Flickr8k dataset has 5 captions associated with it. We used a dictionary to store the name of the image as the key and its corresponding 5 captions as the value in the form of a list. This separates the image names from its captions as well.

Splitting into Train, Test and Validation

Next we create 3 separate dictionaries for training, testing and validation purposes and copy our originally created dictionary into these 3.

Adding Sequence Tokens

Next, we add sequence tokens to the text to mark the beginning and end of the caption. The start token's purpose is to let the model know that caption generation should begin, and the end token is to let the model know that caption generation should stop. We added

a “startcap” token as a start token and an “endcap” token as an end token to each caption by iterating in the image names (dictionary’s keys) and making changes to the value list.

Caption maximum length computation

We proceed to calculate the maximum length of caption. This helps in defining the input size for training, to ensure that every caption is of constant length for better computations. Also, knowing the maximum length helps in determination of the amount of padding or truncation point as well as helps in prevention of excessively long caption generation. Thus, we calculated the maximum caption length via a simple for loop by updating the max value whenever a caption size greater than current max is encountered.

Calculating vocabulary size

Next we find out the vocabulary of our dataset as well as it’s size. The vocabulary size is the number of unique/distinct words present in all the words in all captions of the dataset. A larger vocabulary size indicates a better, diverse dataset. Computing vocabulary size will help us in calculating the word embedding matrix that we will see below. We iterate through the images in the dictionary and check for every caption’s word has occurred previously or not, if no, then the word gets added to a list called **vocab_list** which is a list of all distinct words. The **vocab_size** is given by len(vocab_list).

WORD EMBEDDINGS GENERATION

Now we proceed towards word embeddings generation. One hot encodings are quite memory intensive and aren’t suitable for large vocabularies. We use **GloVe embeddings** which generate dense vector representations, where words are bridged based on how frequently they occur simultaneously. Each word is embedded to a fixed dimension whose length is equal to the size of the vocabulary. We create a GloVe embedding text file with each line in the file containing : the word itself followed by it’s 200 dimensional embedding. Next for every word present in the glove embedding, we add it’s corresponding embedding to a vocabulary dictionary called **glove_emb_dict**

EMBEDDING MATRIX CREATION

Now for creating an embedding matrix, we will iterate through our own vocabulary list **vocab_list**, check if the same word exists in the glove embeddings dictionary **glove_emb_dict** and use the list's word index as index for our embedding matrix as well. We first initialized a random matrix using **np.random.uniform** and then proceeded towards filling the matrix with word embeddings.

VISUAL FEATURE EXTRACTION

CNN - Model VGG16

We import the inbuilt VCG-16 (Visual Geometry Group 16) model in Keras which is a deep neural network CNN with 16 layers with 13 convolutional layers and 3 fully connected layers enabling it to capture hierarchical features of images. It returns a **4096** dimensional feature representation which can be used in our LSTM.

LANGUAGE FEATURE EXTRACTION

We convert the one-hot representations of the words in a caption into word embeddings using the embedding matrix.

COMBINING THE MODALITIES and PASSING THROUGH LSTM BLOCK

Next we combine the image feature vector that we obtained after the visual feature extraction step with the sequence of word embeddings. The resultant sequential data is then passed through a 3-layered LSTM network with DropOut layers in between. We take the output of the last time stamp of the third layer of the LSTM block and combine it with the feature vector of the image. The resultant vector is then passed through a decoder block consisting of fully connected layers and softmax layer at the end to generate the

posterior probability vector. During training, we backpropagate after calculating cross entropy loss using this posterior probability vector. During inference, we select the word from the vocabulary which has the maximum probability in the posterior probability vector.

BUILDING THE DATA-LOADER

We implemented a method called **data_generator()** which takes the image features, caption and batch_size to generate batch_size number of training samples at a time. This dataloader will thus enable faster access of data during training, and help in parallelizing data loading which optimizes memory usage, making use of multiple GPU's thereby reducing time spent on waiting for data.

TRAINING THE CNN-LSTM SYSTEM

We used the following hyperparameters for training the entire CNN-LSTM network.

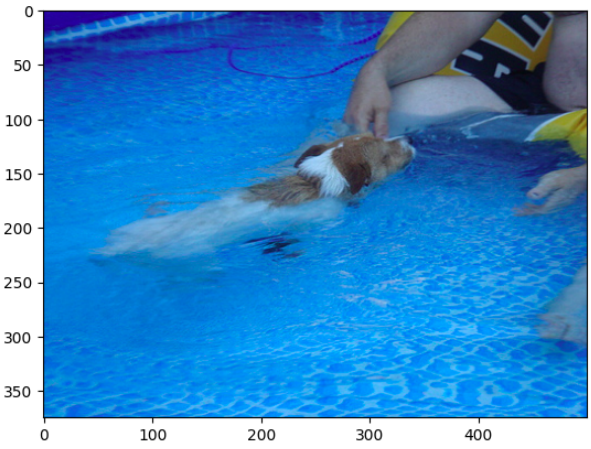
HYPERPARAMETER	VALUE
Number of epochs	50
Batch Size	160
Steps per epoch	len(train_dict)*5 // (batch_size) = (6000*5) // (160) = 187

We then saved our trained model as a pickle file, ready for deployment so that the trained weights are saved.

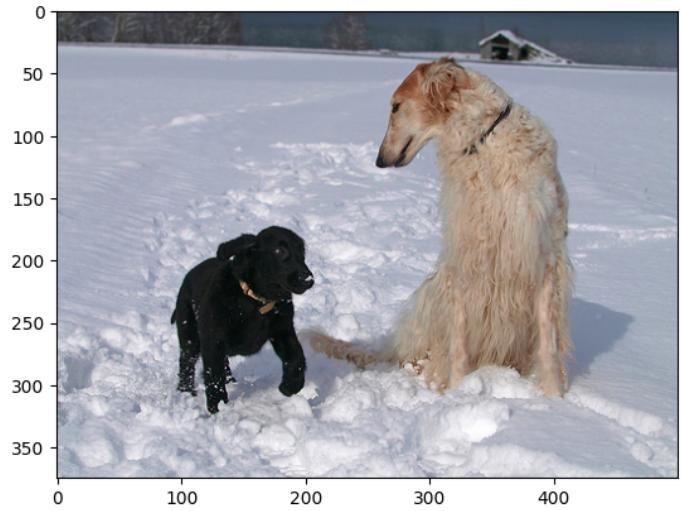
TESTING THE CNN-LSTM SYSTEM

We then proceeded to test the model on various random test images. We implemented a function called **get_caption()** that takes the image as an argument and calls our trained model to predict the caption for the image passed, removes the start and end tokens and returns the final prediction.

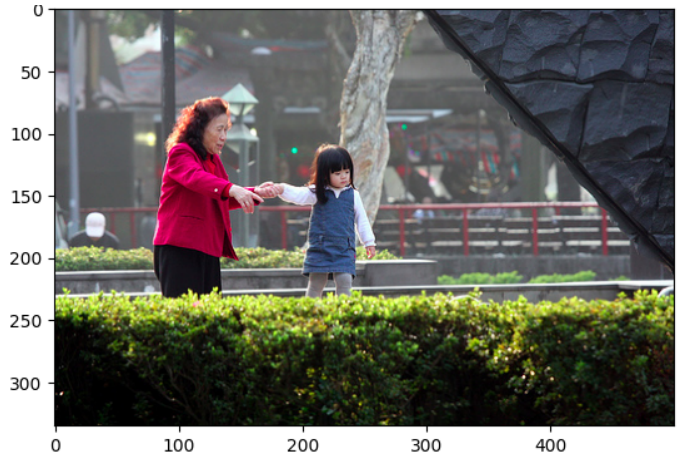
RESULTS & EXAMPLES OF PREDICTIONS



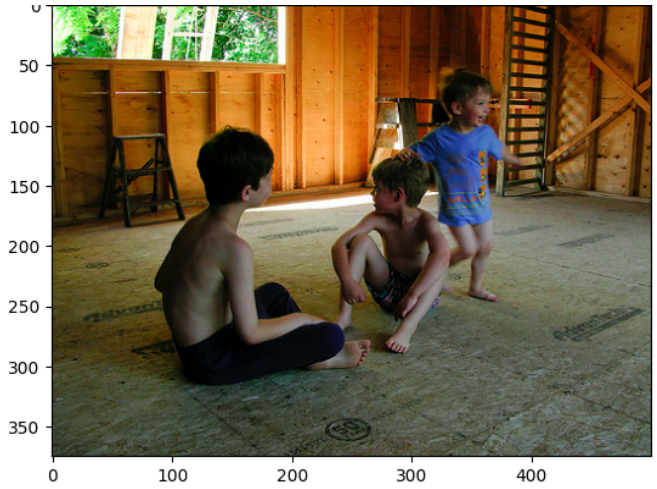
A dog is laying on a rock wall .



Two black dogs play in the snow .



A girl is sitting on a bench in front of a large crowd of people .



A man is petting a man in a grey shirt .

EVALUATION - BLEU & METEOR SCORE

BLEU SCORE COMPUTATION

BLEU (Bilingual Evaluation Understudy) score is a metric to evaluate the quality of NLP tasks. It compares the predictions with one or more references and checks for the similarity between them. BLEU score value is a floating point number in between 0 and 1, higher the

score, greater the similarity with the reference. It works based on **N-gram precision** where it computes the number of overlapping N-grams and divides it by the total number of N-grams. We computed the bleu score for our baseline model by calling the **sentence_bleu()** method from the **nltk library**, and the results are as follows:

METHOD	BLEU SCORE VALUE
Average	0.55934
Maximum	1.0
Mode	0.4768
Median	0.5477

METEOR SCORE COMPUTATION

Next we proceed towards calculation of the meteor score (Metric for Evaluation of Translation with Explicit ORdering) which is another metric for evaluation of NLP tasks. It takes into account **1-gram precision** i.e single word overlaps, **recall** and **harmonic mean**. Thus meteor score captures even the conceptual aspects of predictions. Meteor score also, like BLEU score lies between 0 and 1, higher the score, higher is the similarity between reference and predictions. Thus we computed the meteor score by calling the **meteor_score()** function from the **nltk library**. The results are as follows:

METHOD	METEOR SCORE VALUE
Average	0.3842
Maximum	1.0
Mode	0.2976
Median	0.35898

2. MODIFIED BASELINE MODELS

For the Modified Baseline Models, the preprocessing part and the architecture of LSTM remains the same. We have overall created 4 Modified Baseline Models.

2.1 *MODIFIED VERSION 1* - Altered Vision Embedding

In this version of modified baseline, we modified our Visual representation via usage of the Resnet50 CNN model instead of the VCG16 CNN model. Following is a description of the ResNet50 model.

CNN - Model ResNet 50

ResNet-50, a powerful deep convolutional neural network (CNN) imported into Keras, has a depth of **50 layers**. Its capability to capture intricate visual features results in the generation of a feature representation with a size of **2048**. The architecture of ResNet-50 is constructed using repetitive blocks, which include convolutional layers, rectified linear unit (ReLU) activations, and batch normalization layers. Through the utilization of these blocks, ResNet-50 extracts **hierarchical representations** from images, incorporating both local and global visual information. The network's significant depth plays a role in delivering performance and accuracy in various vision tasks, making it a reliable option for a wide range of applications.

The evaluation of this version yielded the following results for BLEU and METEOR scores:

BLEU & METEOR Scores: VERSION-1

METHOD	BLEU SCORE VALUE
Average	0.565496
Maximum	1.0
Mode	0.48701
Median	0.5595

METHOD	METEOR SCORE VALUE
Average	0.38884
Maximum	0.97441
Mode	0.16666
Median	0.3613

Here we can see that we have obtained higher BLEU and METEOR scores as compared to the baseline model. This was expected, as in this version of modified baseline we have used the ResNet50 model as an image feature extractor instead of the vgg16 model. The ResNet50 model is a more sophisticated CNN backbone for feature extraction as compared to vgg16. Thus the ResNet50 model gave us improved visual embeddings, thereby giving the “modified baseline version 1” an edge over the baseline model.

2.2 *MODIFIED VERSION 2* - Single Head Attention

In this version of the modified baseline, we add single head attention. We are still using the ResNet50 visual feature extractor as used in “Modified Version-1”. The architecture is as follows:

We compute the word embeddings using our embedding matrix and then pass them through a fully connected layer. Next we concatenate the visual feature vector obtained from the pre-trained ResNet50 model to the sequence of word embeddings.

Next we compute the query, value and key vectors using learnable query, value and key weight matrices. These are used to compute single head attention and produce enhanced embeddings both corresponding to vision and language modalities.

Since the image feature vector was concatenated to the sequence of word embeddings, during the computation of attention, the word embeddings are going to be enhanced by keeping in mind the context of the image.

The sequential data of improved embeddings are passed onto the LSTM and decoder block which computes the posterior probability vector.

The evaluation of this version yielded the following results for BLEU and METEOR scores:

BLEU & METEOR Scores: VERSION-2

METHOD	BLEU SCORE VALUE
Average	0.56397
Maximum	1.0
Mode	1.0
Median	0.5598

METHOD	METEOR SCORE VALUE
Average	0.3883
Maximum	0.9925
Mode	0.25
Median	0.3611

By looking at the Average BLEU and METEOR score results we can see that the “modified baseline version 2” is certainly a better model than the baseline model. This was expected because here we not only have the sophisticated ResNet50 model as a visual feature extractor, but also have single head attention module which further improves the word embeddings keeping in mind the context of the input image.

Also, if we look at the mode of the BLEU and METEOR scores we would notice that this single attention head version of the modified baseline is better than the “modified baseline version 1”, because most of the times the single head attention version of the modified baseline gives much better results. This was expected because here we have additionally used single head attention which has the capability to attend to specific regions of the sequential data and improve the embeddings.

2.3 *MODIFIED VERSION 3* - Multi Head Attention

BLEU & METEOR Scores: VERSION-3

In this version of our modified baseline we replace single head attention with multi-head attention. This way we will be able to attend to multiple locations of the sequential data simultaneously with the help of multiple heads. We combine the results of all the heads using a linear layer to get the final improved embeddings. These are then passed onto the LSTM and decoder block as usual.

We have performed experiments using different values of number of attention heads. The results obtained are as follows:

Number of Attention Heads	Metric	Value
2	Average BLEU Score	0.56824
2	Average Meteor Score	0.38582
12	Average BLEU Score	0.56346
12	Average Meteor Score	0.39198

8	Average BLEU Score	0.5722046
8	Average Meteor Score	0.39341

Here we have obtained the highest average BLEU and average METEOR score. This was expected as here we have used the ResNet50 visual feature extractor and have replaced single head attention with multi-head attention. Thus we are able to attend to multiple parts of the sequential data as required simultaneously.

2.4 *MODIFIED VERSION 4* - Beam Search

The beam search algorithm selects the most likely sequence of words that maximize the probability of the entire sequence.At each step, the algorithm generates all possible next words for each partial sequence and calculates the probability of each resulting complete sequence. The algorithm then selects the **top-K** most likely complete sequences from all the possible combinations of the partial sequences and their next-word candidates.

In our case, we have taken **K to be 3.**

BLEU & METEOR Scores: VERSION-4

METHOD	BLEU SCORE VALUE
Average	0.4862
Maximum	0.8743
Mode	0.3421
Median	0.4925

METHOD	METEOR SCORE VALUE
Average	0.2853
Maximum	0.6626
Mode	0.157

Median	0.2701

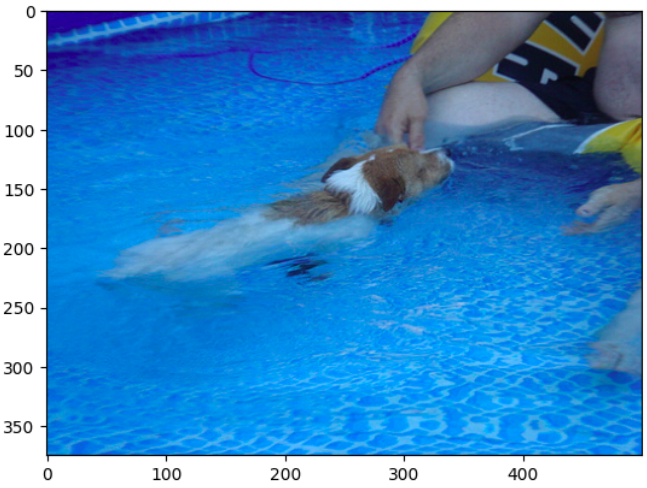
ANALYSIS OF RESULTS

Thus using attention mechanisms in image captioning has shown improvement in the quality of generated captions. This is mainly because attention allows the model to selectively focus on particular image regions during the caption generation process. By using attention, the model gains the capability to direct its focus to different aspects of visual information, leading to enhanced performance in caption generation.

In image captioning, the model aims to understand the meaning of an image and generate a descriptive caption in natural language. Attention mechanisms help by allowing the model to focus on different parts of the image, giving more importance to the regions relevant to the words being generated. This selective attention improves the model's capacity to capture relevant visual details, resulting in more accurate and contextually suitable captions.

For the Beam Search part, it took a lot of time for the results to be analyzed. Thus, we just checked the BlUE score for **first 100** test images only. The results were not as satisfying as previous attention algorithms mainly due to time constraints and limited testing.

MODIFIED BASELINE PREDICTION EXAMPLE



A little boy is laying on a blue and blue plastic pool .

Best Modified Version

REFERENCES

We referred the following for our better understanding of the concept and as a guide to be able to complete this project:

1. https://towardsdatascience.com/a-guide-to-image-captioning-e9fd5517f350
2. https://towardsdatascience.com/image-captioning-in-deep-learning-9cd23fb4d8d2#:~:text=What%20is%20Image%20Captioning%3F&text=Image%20Captioning%20is%20the%20process,Vision%20to%20generate%20the%20captions.