VR Mini Project - 2

Done by:

IMT2018002 - Abhigna Banda
IMT2018026 - Gayathri V.
IMT2018046 - Mundla Aarthi Sree
IMT2018047 - Nachiappan SK

Team Code: AGNA

Problem Statement

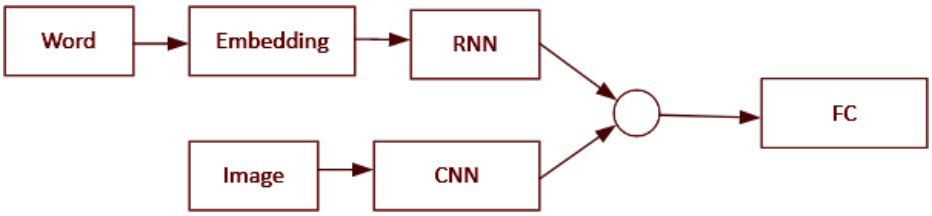
To design a CNN-LSTM system that can perform image captioning on the Flickr8K dataset.

Introduction to CNN and LSTM

CNN-LSTM is short for the CNN Long Short-Term Memory Network. It is a special kind of LSTM architecture specifically designed for sequence prediction problems with spatial inputs, like images or videos. This architecture combines convolutional neural network layers for feature extraction, along with LSTM to support sequence prediction.

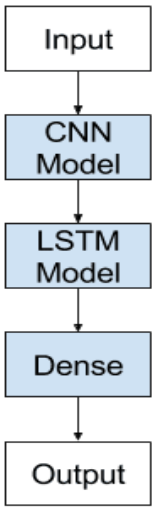
CNN LSTMs were designed specifically for problems of these kinds:

- Activity Recognition: Generating a textual description of an activity demonstrated in a sequence of images.
- Image Description: Generating a textual description (caption) of a single image.
- Video Description: Generating a textual description of a sequence of images.



This architecture is appropriate for problems that:

- Have **spatial structure** in their input, like the 2D structure or pixels in an image or the 1D structure of words in a sentence.
- Have a **temporal structure** in their input such as the order of images in a video or words in text, or require the generation of output with temporal structure such as words in a textual description (like image captioning).

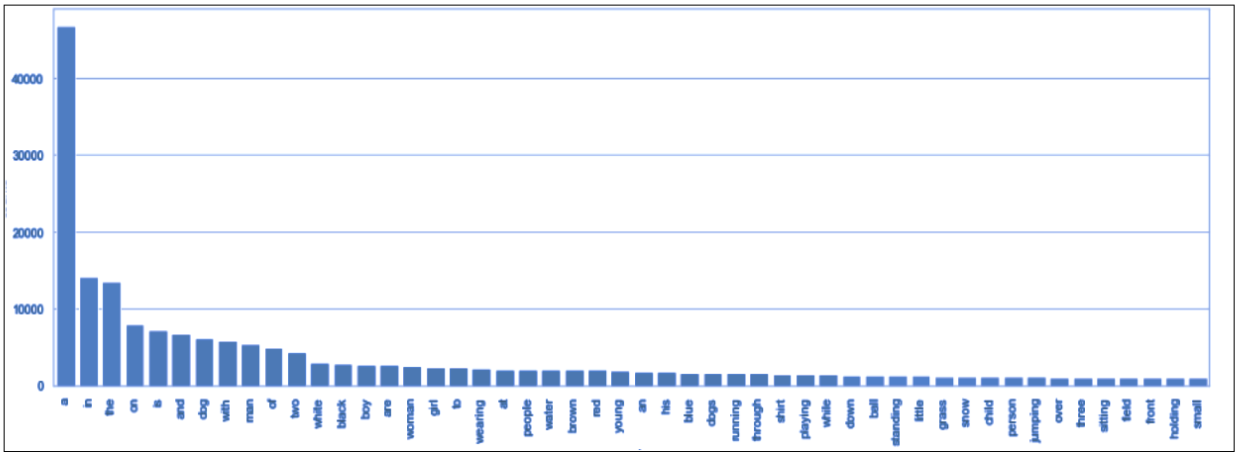


Dataset Description

The Flickr8k dataset consists of 8000 images, each image of which contains 5 different captions. Due to its small size, the dataset can be used to easily train on low-end systems.

As a **data-cleaning** step, we can remove all the words with very less frequency (<1% of the total vocabulary size), as they are likely to play an insignificant role in the caption generation. On plotting the words based on frequency of occurrence, we obtained the following:-

Plot of top 50 words:



Each caption has been encoded as follows:

<start> + caption + <end>

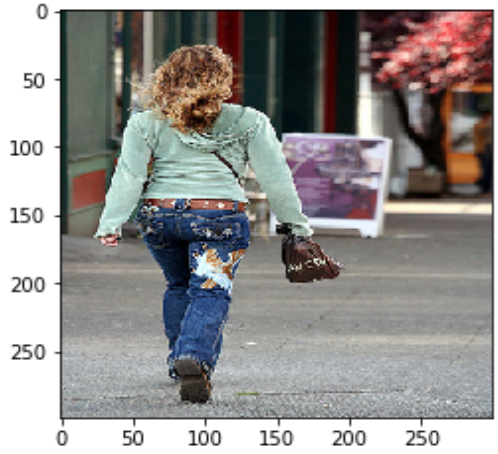
When the model encounters <start>, the caption generation begins, and it will stop predicting as soon as it encounters the <end> symbol.

Processing the Data

We will predict the caption word by word. Thus, we need to encode each word into a fixed size vector. For this, we create two dictionaries holding a mapping between word to index, and vice versa (every word is assigned a unique index to make computations easier).

We have 1652 unique words in the vocabulary, and thus each word will be represented by a unique integral index between 1 to 1652.

Example Image:



Caption = “A girl in a blue hoodie is carrying a brown bag”

Partial Caption	**Target word**
<start>	A
<start> A	girl
<start> A girl	in
<start> A girl in	a
<start> A girl in a	blue
<start> A girl in a blue	hoodie
<start> A girl in a blue hoodie	is
<start> A girl in a blue hoodie is	carrying
<start> A girl in a blue hoodie is carrying	a
<start> A girl in a blue hoodie is carrying a	brown
<start> A girl in a blue hoodie is carrying a brown	bag
<start> A girl in a blue hoodie is carrying a brown bag	<end>

- For every image, we construct the image feature vector, partial captions and the target word corresponding to each caption.
- We represent the partial captions as a vector of size equal to the maximum caption length in the training dataset. Since we cannot perform any arithmetic with strings, we map each word to a

corresponding numerical index and use the same for representational purposes. All the elements in the partial caption vector that have not been assigned a value default to 0.

Partial Caption	**Target word**
[1, 0, 0 …. 0]	5
[1, 5, 0 …. 0]	6
[1, 5, 6 …. 0]	3
[1, 5, 6, 3 …. 0]	2
[1, 5, 6, 3, 2 …. 0]	7
[1, 5, 6, 3, 2, 7 …. 0]	8
[1, 5, 6, 3, 2, 7, 8 …. 0]	4
[1, 5, 6, 3, 2, 7, 8, 4 …. 0]	9
[1, 5, 6, 3, 2, 7, 8, 4, 9 …. 0]	12
[1, 5, 6, 3, 2, 7, 8, 4, 9, 12 …. 0]	10
[1, 5, 6, 3, 2, 7, 8, 4, 9, 12, 10 …. 0]	11
[1, 5, 6, 3, 2, 7, 8, 4, 9, 12, 10, 11 …. 0]	13

Measure of Accuracy

We have used BLEU scores to determine the efficacy of our model. BLEU stands for Bilingual Evaluation Understudy Score. It is an algorithm used to evaluate the quality of machine generated text as compared to the ground truth. We can use BLEU to check the quality of our generated caption.

- BLEU is easy to understand and compute.
- It lies between [0,1]. Higher the score, more accurate the caption.

Calculating BLEU Scores:

BLEU Score is calculated by finding the maximum number of times the n-gram appears in the reference divided by the number of times it appears in the prediction. Here n-gram can be either unigram or bigram.

Example:

Predicted Caption: There are birds in a tree.

Reference: There are two birds sitting in a tree.

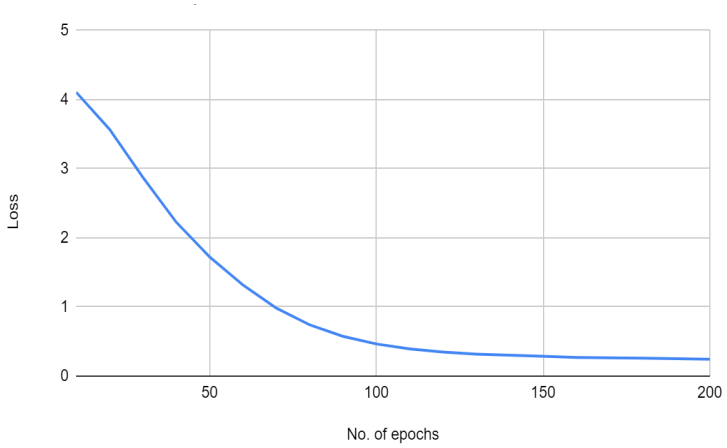
Bigram(predicted) = (there, are), (are, birds), (birds, in), (in, a), (a, tree)
Bigram(reference) = (there, are), (are, two), (two, birds), (birds, sitting), (sitting, in), (in, a), (a, tree)

Therefore, BLEU score = 1/5 + 0/5 + 0/5 + 1/5 + 1/5 = 3/5

Values Obtained

No. of epochs	Accuracy in Sigmoid	Accuracy with ReLU
1	3.05%	7.73%
50	33.27%	48.37%
100	65.80%	79.32%
150	74.61%	83.24%
200	78.46%	85.13%
250	80.98%	85.68%

Plot for no. of epochs vs loss



Batch Normalization and Dropouts

Fixing the number of epochs as **200**, we obtained the following BLEU scores with dropouts and batch normalization:

Condition	BLEU Score
Without dropout	0.304
With dropout = 50%	0.489
Without batch normalization	0.491
With batch normalization, dropout	**0.518**

Learning Algorithms

Algorithm	BLEU Score
SGD	0.407
Adam	**0.518**

Model Architecture

To extract image features we have used **InceptionV3** and **ResNet50**. We have chosen these because:

- They take less memory (in the range of 90-100 MB), and
- Less parameters (approximately 25 million parameters or less)

- The model takes in two inputs:
    a. The output of the RNN which takes in the partial caption as input, and
    b. The image vector
- The model uses activation functions like ReLU and softmax to return two quantities:
    a. Predicted word (based on maximum probability distribution)
    b. Next word for the current partial caption.

Flow of the partial caption:

- The input layer takes in an array of indices corresponding to partial captions.
- Every index in this partial caption gets mapped to a 200-dimensional feature vector in the embedding layer.
- We have used **dropout layers** to prevent overfitting of the model.
- This is then passed to the LSTM. This will help the model learn about the 'ordering' of words in a sentence.

Flow of image vector:

- The input layer thales in a feature vector of dimension 2048.
- There is another dropout layer to minimise overfitting, which is then passed to the dense layer.

At the end of these two steps that run parallely, the output of both are of dimension (batch size, 256), and are hence merged into a single vector.

The output layer uses **softmax** to generate probability distribution across all the words in the vocabulary, based on which the final prediction will be made.

Model Description:

- Total params: 1,483,648
- Trainable params: 1,483,648
- Non-trainable params: 0

The following were used:

- Activation Function: ReLU
- Regularization: 50% dropout and batch normalization
- Hyperparameters:
    - Learning Rate: 0.01
    - Batch Size: 100
    - Number of epochs: 200
- Loss Function: Categorical Cross-Entropy
- Optimizer: Adam

Results

The following images show the captions predicted by our model:

A newly taking this to to pink shirt is chases a picture above a beach .

A group of men sit in rocks pigeons .

A bunch of people look , at indoor stool .

A man is wearing an 3 jacket riding three bike down his white white white chair down three group .

A man is is sitting on a snowy mountains .

A group of people are and are set at in a , small wave on a ocean .

A child are in a face paint in a white white brown brown blowup

A young girl wearing a pink tank shirt jumping on a slide .

A little boy is swinging on a pole at the background .

A man wearing a red and reading a woolen .

Therefore**, Final BLEU score on the Flickr 8k dataset : 0.5184**

Scope for Improvement

- The training time of the model is over 1 hour. This needs to be improved by making computations more efficient.
- The final BLEU value is 0.518. The accuracy of the model can be further improved by training it on a larger dataset and hyperparameter tuning.
- Experimentation with more hyperparameter tuning, like learning rate, batch size, number of layers, number of units, dropout rate, batch normalization etc can help us better understand the model and improve its performance.
- Using cross-validation to prevent overfitting.
- Unigram BLEU is better for shorter sentences while bigram is better for longer sentences. It is important to take both into account for accurately gauging the performance of the model.