# Assignment 3

Group No 20

Tanay Dixit (EE19B123) , Atharva Chougule (CS19B016) , Kaustubh Miglani (CS19B060)

3th May , 2022

CS6910

# 1 Image Captioning

## 1.1 Task1:Using RNN based decoder for Image Captioning

### Generating the Dataloaders

We were given a dataset of 4000 images and 5 captions for each of these 4000 images, We first built a vocabulary for the words appearing in the captions and assigned each word a unique index. The words `'<pad>'`,`'<start>'`,`'<end>'` were added to the vocabulary list which signify the padding , stat and end of caption respectively. We then proceeded to embed each if this word into a numerical representation. We have used `GloVe` for generating embeddings for captions. For this we used the Stanford's `GloVe` pre-trained word vectors 'glove.6B.300d' which has a 300 dimension representation for 40K different english words. The words of the vocabulary which were not a part of the pre-trained `GloVe` word vectors were initialized to random vectors of 300 dimension. To each of this 300 dimension vector we added one more feature to distinguish between normal words and special i.e star,end and pad words. Hence our word embeddings are of size 301. After this we proceeded with loading our dataset.
We have used a train-test split of 0.8:0.2 as mentioned in the assignment.

### Model Description

The model comprises of an encoder part and a decoder part. The encoder part is used extracting feature vectors from the image. The Decoder uses this image vector to generate captions for the image.

### Encoder

1. We have used the pretrained Resnet50 model as the first part of our Encoder.
2. We have removed the last 2 layers of Resnet and used a NetVlad layer to further learn the representation of images in a better way.
3. Finally we use a linear layer which has ouput dimension of 4096.
4. Hence the Encoder transforms the given images into a feature vector of dimension 4096.

### Decoder

1. The Decoder Consists of a single layer RNN. The dimension of hidden layer of RNN is equal to 4096 which is the dimension of image feature vector which encoder generates. The input layer dimension of RNN is 301 which is the dimension of word embeddings.

2. The Decoder also comprises of a linear layer whose output dimension is equal to vocabulary size of the data. The max value in this output layer of size equal to vocab size gives what word is predicted by the RNN. (During prediction phase we use this word as next input word for RNN.)

### Training the Model

During the training phase the input for RNN is the actual caption that has been provided for the training images, while the target output of the RNN is the same caption shifted 1 word to the right. Thus we train the RNN to predict the next word which is fed the current word as the input. The hidden state of the RNN is initialized to the encoded image feature vector.
We used Cross Entropy Loss as the Loss function for the Model. We also used Adam's method as the optimization technique to speed up the learning process.
The loss curve during training are as follows (Figure1).
The final loss was found to be 2.783

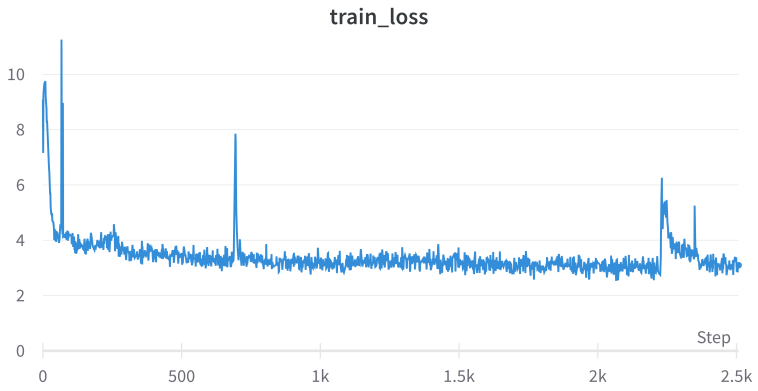



*Figure 1: Training Curves*

### Prediction Phase

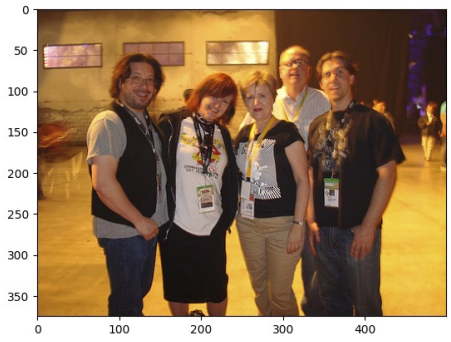
Now we use the train model to find predict the captions of images from the test data. During prediction phase we don't use the gold truths like we used in training, we send only the `<start>` word as the initial input and the predict the next word, this predicted word is now fed as input to the predict the word that comes after it, this process repeats until the `<end>` word is generated which specifies the end of caption. After the prediction was done we calculated BLEU score of the generated captions with the given caption for the test data. The BLEU scores are as follows. We use the following variant of the BLEU metric:

| Bleu-1 | Bleu-2 | Bleu-3 | Bleu-4 |
|---|---|---|---|
| 0.462 | 0.087 | 0.026 | 0.008 |

*Table 1: BLEU Scores for Image captioning using RNN*

to compute BLEU-N, we make use of **only** `n-gram` overlaps. We observe that BLEU-1 has the highest score followed by BLEU-2 and then BLEU-3 ,BLEU-4.This also makes sense as BLEU-1 only involved 1-gram overlaps which is simple when compared to 4-gram overlap in BLEU-4.
We also analyse the model's predictions qualitatively and present some samples as follows:

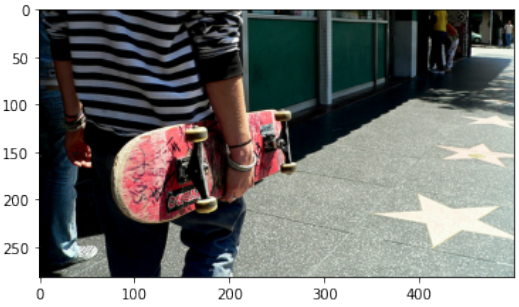
### Some of the best predictions of RNN



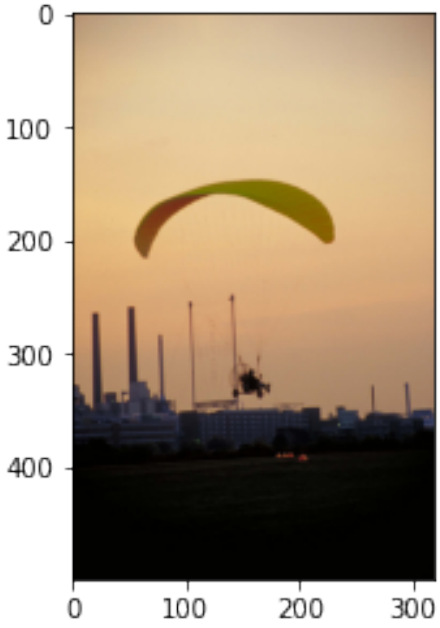

*(a)* ***Predicted Caption****: a group of people gather on a beach .*


*(b)* ***Predicted Caption****: a woman in blue shirt stands on yellow .*

**Some of the worst predictions of RNN**




*(a)* ***Predicted Caption****: a little boy is running across the beach .*




*(b)* ***Predicted Caption****: a kid jumps across the beach .*

**Observations**

The observations regarding this model are:-

- The training loss does not go below 2. We can attribute this problem to the problem of vanishing gradients in case of RNN and as we can see from the Training curve as well, the training loss decreases very slowly as the steps keep on increasing.
- The model can often identify some basic subjects (For example:- little boy, woman , group of people) but often fails to identify the action in the given image .
- The captions often end with the same set of words (For example :-across the beach) and this points to the problem of not being able to identify the action properly.
- The best predictions by this model are average and sometimes completely unrelated captions are generated by the model.

- All of these problems can be in part attributed to the problem of vanishing gradient problem seen in case of RNN as the model is not able to learn more intricate details of image and converting it to captions . And as we will be seeing later in case of LSTM this problem is avoided and therefore LSTM will be performing better than RNN.

## 1.2 Task2: Using LSTM based decoder for Image Captioning

### 1.2.1 Generating the Dataloaders

We were given a dataset of 4000 images and 5 captions for each of these 4000 images, We first built a vocabulary for the words appearing in the captions and assigned each word a unique index. The words `'<pad>'`,`'<start>'`,`'<end>'` were added to the vocabulary list which signify the padding , stat and end of caption respectively. We then proceeded to embed each if this word into a numerical representation. We have used `GloVe` for generating embeddings for captions. For this we used the Stanford's `GloVe` pre-trained word vectors 'glove.6B.300d' which has a 300 dimension representation for 40K different english words. The words of the vocabulary which were not a part of the pre-trained `GloVe` word vectors were initialized to random vectors of 300 dimension. To each of this 300 dimension vector we added one more feature to distinguish between normal words and special i.e star,end and pad words. Hence our word embeddings are of size 301. After this we proceeded with loading our dataset.
We have used a train-test split of 0.8:0.2 as mentioned in the assignment.

**Model Description**

The model comprises of an encoder part and a decoder part. The encoder part is used extracting feature vectors from the image. The Decoder uses this image vector to generate captions for the image.

**Encoder**

1. We have used the pretrained Resnet50 model as the first part of our Encoder.
2. We have removed the last 2 layers of Resnet and used a NetVlad layer to further learn the representation of images in a better way.

3. Finally we use a linear layer which has output dimension of 4096.
4. Hence the Encoder transforms the given images into a feature vector of dimension 4096.

**Decoder**

1. The Decoder Consists of A single layer LSTM. The dimension of hidden layer of LSTM is equal to 4096 which is the dimension of image feature vector which encoer generates. The input layer dimension of LSTM is 301 which is the dimension of word embeddings.
2. The Decoder also comprises of a linear layer whose output dimension is equal to vocabulary size of the data. The max value in this output layer of size equal to vocab size gives what word is predicted by the LSTM. (During prediction phase we use this word as next input word for LSTM.)

**Training the Model**

During the training phase the input for LSTM is the actual caption that has been provided for the training images, while the target output of the LSTM is the same caption shifted 1 word to the right. Thus we train the LSTM to predict the next word which is fed the current word as the input. The hidden state of the LSTM is initialized to the encoded image feature vector.
We used Cross Entropy Loss as the Loss function for the Model. We also used Adam's method as the optimization technique to speed up the learning process.
The loss curve during training are as follows (Figure4).
The final loss was found to be 0.1671

**Prediction Phase**

Now we use the train model to find predict the captions of images from the test data. During prediction phase we don't use the gold truths like we used in training, we send only the ¡start¿ word as the initial input and the predict the next word, this predicted word is now fed as input to the predict the word that comes after it, this process repeats until the


*Figure 4: Training Curves*

¡end¿ word is generated which specifies the end of caption. After the prediction was done we calculated BLEU score of the generated captions with the given caption for the Test data. The BLEU scores are as follows. We use the following variant of the BLEU metric:

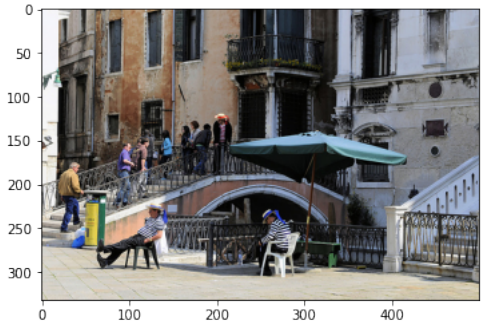
| Bleu-1 | Bleu-2 | Bleu-3 | Bleu-4 |
|---|---|---|---|
| 0.548 | 0.202 | 0.077 | 0.031 |

*Table 2: BLEU Scores for Image captioning using LSTM*

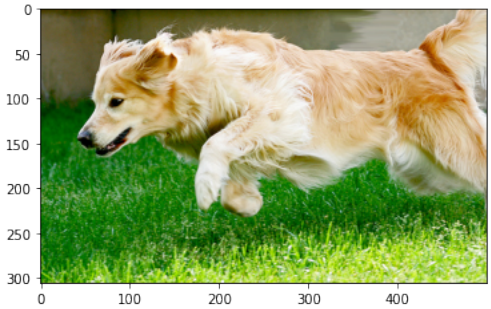
to compute BLEU-N, we make use of **only** `n-gram` overlaps. We observe that BLEU-1 has the highest score followed by BLEU-2 and then BLEU-3 ,BLEU-4.This also makes sense as BLEU-1 only involved 1-gram overlaps which is simple when compared to 4-gram overlap in BLEU-4.

We also analyse the model's predictions qualitatively and present some samples as follows:

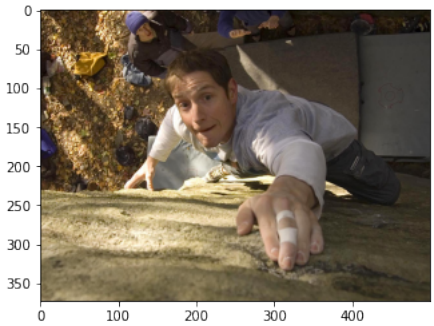
**Some of the best predictions of LSTM**



*(a)* ***Predicted Caption****: a man sits on a stage near a city street .*



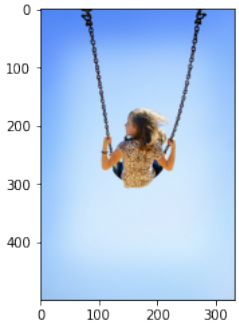
*(b)* ***Predicted Caption****: a white dog running through the grass .*



*(a)* ***Predicted Caption****: a man in a gray shirt and green shorts is climbing up a rock .*

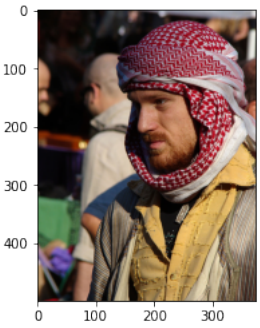
*(b)* ***Predicted Caption****: a man surfs in the ocean.*

*(a)* ***Predicted Caption****: a dirt biker flies through the air .*



*(b)* ***Predicted Caption****: a young girl is riding a swing high in the air .*

**Some of the worst predictions of LSTM**





*(a)* ***Predicted Caption****: a man with a bushy beard and a hat , stands on his cellphone .*

*(b)* ***Predicted Caption****: a man is standing on the grass next to a pond .*

### 1.2.2 Observations

1. We can see that LSTM performs very well in captioning the images it gave us a very formidable train error of 0.1671.
2. Some of the images are captioned almost accurately in case of LSTM , while even the images in which the LSTM performs poorly doesnt have completely wrong captions.
3. Even in the images where the LSTM is poor has atleast somewhat caption related to the image.
4. Also From the Training curve we can see that, in the case of LSTM the vanishing gradients problem don't arise and the error decreases to acceptable values.

**LSTM vs RNN comparision for Image Captioning**


1. The train error in LSTM was 0.1671 which was a significant improvement over RNN based model in which the error didn't even reduce lower than 2.
2. As we can see from the best predictions from above , LSTM succeeds in captioning some images almost accurately. The captions of some images are almost 100% accurate in case of some images. This was absent in the case of RNN which at the most suceeded in only identifying the subject of the image at the best.
3. Even In some of images where LSTM performs poorly it identifies some of the features in the image correctly, while in RNN the some of the captions didnt even make sense and were completely unrelated.
4. From the training curve comparision above we see that the vanishing gradient problem which occured in RNN didn't occur in LSTM which was what was expected.
5. The BLEU1 score of RNN was 0.462 while of LSTM was 0.548 which is significant inmprovement. The BLEU4 score almost showed an improvement of 4 times in the case of LSTM which had a score of 0.031 while RNN only had a BLEU4 score of 0.008.
6. This significant improvement in BLEU4 score suggests that LSTM is able to predict longer collections of words(4-gram) much more accuately than RNN.

7. Overall we can conclude that LSTM is a much superior model and an upgradation to RNN for image captioning task.

# 2 Machine Translation

In this task we translate English sentences into Gujarati with the help of the encoder decoder architecture consisting of a single layer lstm network. We make use of the similar training and inference setup as discussed before.

## Gd `IndicBERT` embeddings for the the Gujarati sentences.

We make use of the spacy tokenizer to tokenize the English sentences and the BERT tokenizer for Gujarati. We observe the following:

1. Approximately 55% of the words in the English training corpus are not present in the GloVe vocabulary. These words are colonial words are often not even english words (transliterations of hindi words , ex: bhartiya, janta, aap, .. etc).
2. On the other hand as Gujarati uses sentencepiece tokenizer it does not face this issue of having out-of-vocabulary words.
3. Finally our English and Gujarati vocabulary have a size of 179,320 and 22,159 respectively. As majority of the English words do not have a GloVe embedding we have to set the embedding layer of the encoder to trainable. While for Gujarati we freeze it.
4. Several sentences in the training corpus were very long and as a result we had to truncate the sentences to a maximum length of 40 tokens( for both languages). We hypothesis that truncating the sentences will not lead to any loss of information because the mean length of the sentences in the training corpus was way less than 40.

### Training

We train a single layer *LSTM* network on the given dataset. Both the Encoder and Decoder lstms have a hidden state of dimension 300. We experiment with training techniques like *teacher forcing* to help the model converge faster. In teacher forcing the gold inputs are fed into the model at every timestep instead of the previous timestep's prediction. We make use of the Adam optimizer and train the network for 3 epochs and the training curve are as follows Figure 9. We were only able to train it for 3 epochs

due to computational limitation(2.5 hrs per epoch) but we observe that even after 3 epochs the losses were still decreasing and performance can be improved further.

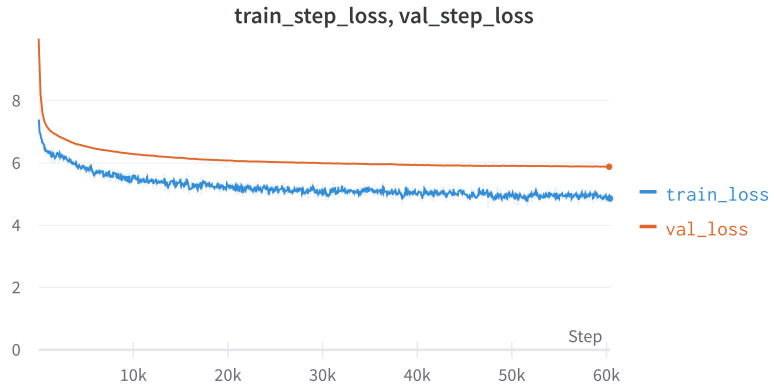



*Figure 9: Training Curves*

We also observe that at every forward pass decaying the teacher forcing parameter provided the best validation results. If only teacher forcing is used we got a validation loss of 6.56 while with our modified method it went down to 5.88.One possible reason for this could be that at start when the model is confused providing the gold input at every time-step (t) helps train the model better and also avoids the errors are previous time-steps to have adverse effects on the later ones($> t$). But as training progresses and the model starts to learn, and providing the model's prediction at timestep $t$ to the input at timestep$(t+1)$ could help it model the sequence better. We use a very modest decay rate of 0.9999.

## Evaluation & Results

**Quantitative Analysis**:We evaluated on our model on the held out test dataset. The BLEU scores are as follows. We use the following variant of the BLEU metric: to compute

| Bleu-1 | Bleu-2 | Bleu-3 | Bleu-4 |
|---|---|---|---|
| 0.074 | 0.003 | 4e-5 | 0 |

*Table 3: BLEU Scores for Machine Translation*

BLEU-N, we make use of **only** `n-gram` overlaps. We observe that BLEU-1 has the highest score followed by BLEU-2 and then BLEU-3 ,BLEU-4.This also makes sense as BLEU-1

only involved 1-gram overlaps which is simple when compared to 4-gram overlap in BLEU-4. Also we get a BLEU-4 score of 0, one possible reason for this could be that the average sentence length for gujarati was around 9 and to get a 4-gram overlap in such cases could be challenging with the given model constraints given.

**Qualitative Analysis**:We also analyse the model's predictions qualitatively and present some samples as follows:

```
SOURCE SENTENCE: my experience after that has been very good .
TARGET: તેના વિશે મારો અનુભવ ઘણો સારો રહયો.
OUTPUT: મારા અનુભવ અનુભવ ખૂબ જ સારો રહ્યો છે.


SOURCE SENTENCE: People of prayagraj are also very enthusiastic about the kumbh .
TARGET: પરયાગરાજના લોકો પણ કુંભ માટે ઘણા ઉતસાહી છે.
OUTPUT: ઉતતરમાંમાંમાં લોકોમાં પણતત પણ પણતહહ ખૂબ જ ઉત્સાહી ..


SOURCE SENTENCE: there is another chapter of history associated with punjab .
TARGET: પંજાબથી જોડાયેલો એક ઇતિહાસ છે.
OUTPUT: પંજાબ સાથે ઇતિહાસ સાથે ઇતિહાસ ઇતિહાસ છે..
```

Overall we feel that some places where improvements could have been made was to use a bi-directional lstm in the encoder as that would help develop a better representation of the input, also using more that 1 layer of lstm would certainly boost performance. We could not experiment with these techniques due to computation and time limitations. We also note that using greedy decoding is not the optimal methods but as other sampling techniques were not taught in class, hence we adhered to greedy decoding. Lastly we observe that BLEU metric is not ideal for measuring the performance of the models as it does not take into account synonymity and penalises all type of "word" errors equally.