CS6910: Fundamentals of Deep Learning

Programming Assignment-3

Image Captioning and Machine Translation

Team Number 19

Aditya Mohan ED19B001 Shreyas Singh ED19B030

Sujay Srivastava ED19B033

Course Instructor:

CHANDRA SEKHAR C

Dept. Of Computer Science

Indian Institute of Technology Madras-600036

TASK 1 Image captioning using a CNN with NetVLAD as encoder and a single hidden layer RNN-based decoder

1. **Model**

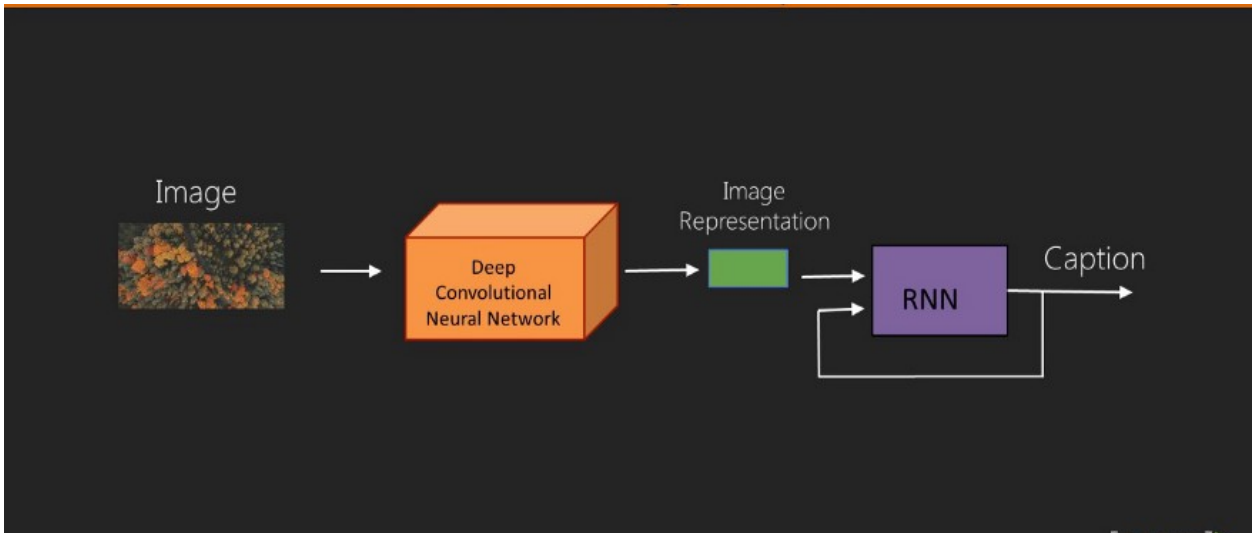


Fig 1 Model used for Image captioning task: CNN with NetVLAD as encoder and RNN-based decoder

Image Representation here is done via NetVLAD

Model Architecture

```
(encoder): EncoderCNN(
  (cnn): VGG(
    (features): Sequential(
      (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): ReLU(inplace=True)
      (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (3): ReLU(inplace=True)
      (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (6): ReLU(inplace=True)
      (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (8): ReLU(inplace=True)
      (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (11): ReLU(inplace=True)
      (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (13): ReLU(inplace=True)
      (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (15): ReLU(inplace=True)
      (16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (17): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (18): ReLU(inplace=True)
      (19): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (20): ReLU(inplace=True)
      (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (22): ReLU(inplace=True)
      (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (24): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (25): ReLU(inplace=True)
      (26): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (27): ReLU(inplace=True)
      (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (29): ReLU(inplace=True)
      (30): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    )
    (avgpool): AdaptiveAvgPool2d(output_size=(7, 7))
    (classifier): NetVLAD(
      (conv): Sequential(
        (0): Conv2d(512, 16, kernel_size=(1, 1), stride=(1, 1))
        (1): Softmax(dim=1)
      )
      (flatten): Flatten()
    )
  )
  (embed): Linear(in_features=8192, out_features=50, bias=True)
)
(decoder): DecoderRNN(
  (rnn): RNN(50, 256)
  (linear): Linear(in_features=256, out_features=400000, bias=True)
  (softmax): Softmax(dim=1)
```

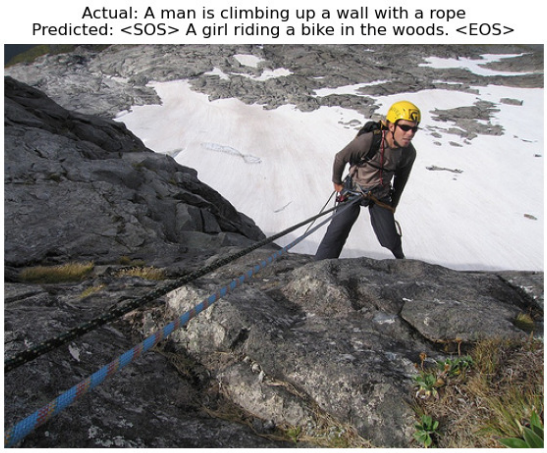
2. Results

BLEU Scores:

BLEU-1 Score	BLEU-2 Score	BLEU-3 Score	BLEU-4 Score
0.3904404136410746	0.0932276603709931	0.0341680419252933	0.0051421544397511

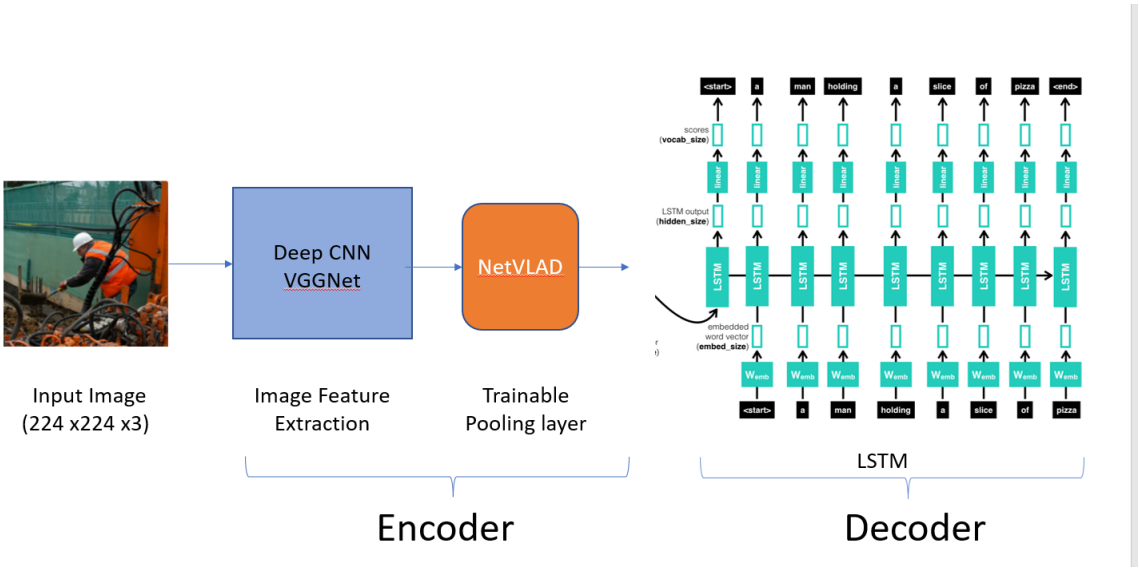
3. Observations

Sample images from the dataset and the outputs:



TASK 2 Image captioning using a CNN with NetVLAD as encoder and a single hidden layer LSTM network-based decoder

1. Model



2. Results

Performance of the image captioning system and the machine translation system is to be given as BLEU@k scores with k = 1, 2, 3, and 4

BLEU-1 Score	BLEU-2 Score	BLEU-3 Score	BLEU-4 Score
0.5103232227190347	0.1452215338512748	0.0351230493246235	0.0140036532646353

3. Observations

Model Architecture

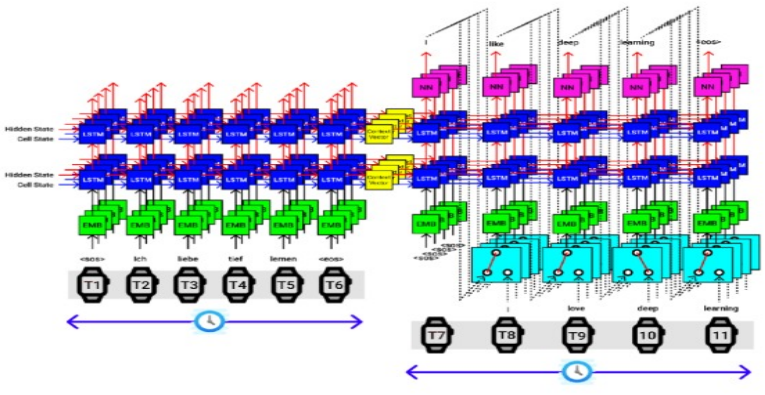
```
(encoder): EncoderCNN(
  (cnn): VGG(
    (features): Sequential(
      (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): ReLU(inplace=True)
      (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (3): ReLU(inplace=True)
      (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (6): ReLU(inplace=True)
      (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (8): ReLU(inplace=True)
      (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (11): ReLU(inplace=True)
      (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (13): ReLU(inplace=True)
      (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (15): ReLU(inplace=True)
      (16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (17): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (18): ReLU(inplace=True)
      (19): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (20): ReLU(inplace=True)
      (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (22): ReLU(inplace=True)
      (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
      (24): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (25): ReLU(inplace=True)
      (26): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (27): ReLU(inplace=True)
      (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (29): ReLU(inplace=True)
      (30): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    )
    (avgpool): AdaptiveAvgPool2d(output_size=(7, 7))
    (classifier): NetVLAD(
      (conv): Sequential(
        (0): Conv2d(512, 16, kernel_size=(1, 1), stride=(1, 1))
        (1): Softmax(dim=1)
      )
      (flatten): Flatten()
    )
  )
  (embed): Linear(in_features=8192, out_features=50, bias=True)
)
(decoder): DecoderLSTM(
  (rnn): LSTM(50, 256)
  (linear): Linear(in_features=256, out_features=400000, bias=True)
  (softmax): Softmax(dim=1)
```

4. Analysis of the Results

- Image Captioning with an LSTM decoder performs better than an RNN-based decoder. It could be attributed to fact that LSTM is computationally more effective. LSTM has a gated unit hidden layer, which enables it to learn longer-term dependencies.
- Image Captioning with LSTM and RNN decoder can form grammatically comprehensible sentences that capture the color and activity being done in the image to decent accuracy
- The model struggles in identifying fine grained details like the gender of the subject , age of the subject , the objects being used etc .

TASK 3 Machine translation with encoder and decoder, each built using a single hidden layer LSTM network

1. Model



Seq2Seq for a batch size of 4. Source — Author

2. Results

BLEU scores:

BLEU-1 Score	BLEU-2 Score	BLEU-3 Score	BLEU-4 Score
0.21122	0.15323	0.017755	0.0011253

3. Observations

Sample English sentence from test data

English: “The states can take up one or two sports of their choice and display their strength.”

After Epoch 1: വൺ ആർഎ ആർഎഞങ്ങളെ സൂര്യൻ വൺ വൺ അബെ(Translated back to English: one *ra ra* us sun one one *abe*)

After Epoch 5: ആളുകൾ അധികാരത്തോട് പോരാടുന്നു, ഒരുതിരഞ്ഞെടുക്കുന്നു

Google Translate: People fight power and one choose

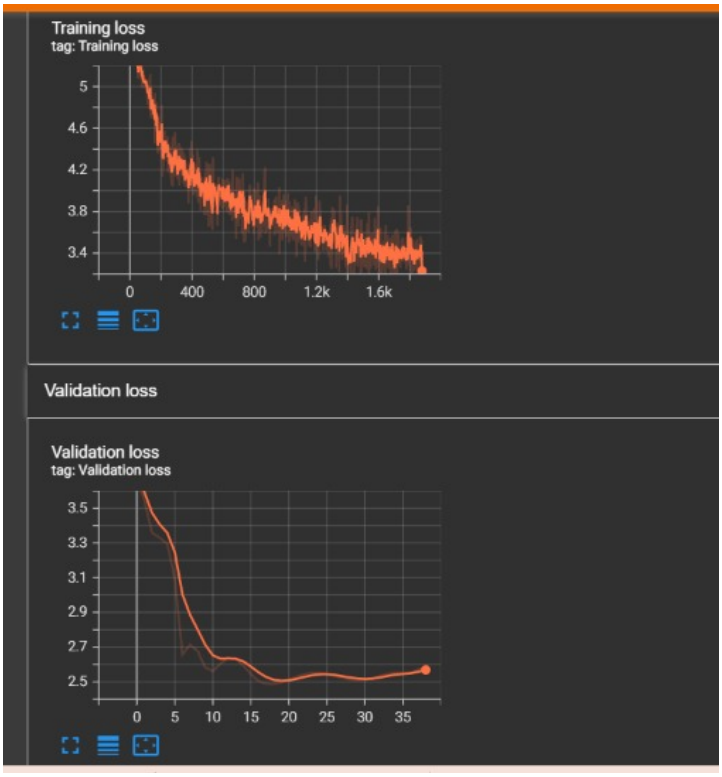
After Epoch 10: അധികാരത്തിൽ ആളുകൾ പന്തും നേരിയ കളിക്കുന്നു ഒന്ന് തിരഞ്ഞെടുക്കുക

(Translation using Google translate: People in power watching the play choose one)

After Epoch 20:

പോരാടാനും ഓപ്ഷനുകളുമുള്ള ആളുകളെ സർക്കാർ തിരഞ്ഞെടുക്കുന്നു ഒന്ന് തിരഞ്ഞെടുക്കുക

Google Translate :Government choose power people to fight and options to choose one



4. Analysis of the Results

1. We ran the model with the following hyperparameters:

    1. Number of training samples = 223728 (10 % of the full dataset)
    2. Number of Validation Samples = 2000
    3. Number of test samples = 2000
    4. Max length of english Vocabulary = 50000

5. Max length of Malayalam vocabulary = 5000

6.encoder embedding size = 100

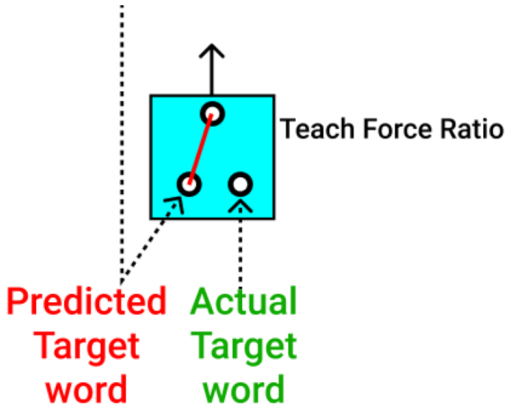
7.decoder embedding size = 128

8.hidden size = 1024

2. We trained the model for 20 epochs with Adam optimizer with a learning rate of 0.001 with a batch size of 32 .

3. The loss decreases slowly from 6.5 and its value is 3.2 after 10 epochs . From the graphs its evident that the model has not converged and its performance can be further improved . We couldn't train further due to time and resource constraints. 1 epoch takes approximately 4 hours to train .

4.Teacher forcing : During training, we randomly pass the true target word to the decoder instead of the predicted target word to add stability during the initial phases of training .



5. The translations produced by our model after 20 are good word-by-word match, but the sentences produced are not fully comprehensible and grammatically correct. We hypothesize that a more complex model and better fine-tuning are required.