CSE 253 Programming Assignment 3

Astuti Sharma
A53285392
asharma@eng.ucsd.edu

Divyanshu Mund
A53281339
dmund@eng.ucsd.edu

Kunal Jain
A53309158
kujain@eng.ucsd.edu

Ria Aggarwal
A53304030
r2aggarw@eng.ucsd.edu

Sai Akhil Suggu
A53284020
ssuggu@ucsd.edu

Abstract

As we grew more curious towards using deep learning for computer vision tasks, we are not satisfied with just finding or recognising objects in an image. We want develop an AI system to give human like description of an image. Hence, came the Image Captioning task. Although at first glance, this might seem elusive, and this is a continuation from our previous project on Semantic Segmentation. Similar to that, we are using Encode-Decoder architecture with encoder(ResNet) to give relevant features of an image and decoder (LSTM RNN) uses these feature to generate captions. In this paper review, we mention our explorations for machine translation of an images from MSCOCO Dataset using Deep Neural Networks. We then explored different modifications of the traditional networks by changing number of layers and network design, showing the advantages and disadvantages of each ablation.

1 Introduction

In our every day life, we observe and perceive information from our surroundings and are able to give a good description of our experience. In fact, we were able to do this from childhood itself. Whereas making an automating system to do this translation of an image is seemed elusive until the development of deep neural networks. Although, we were able to identify and recognise obeject in an image with good confidence, we didn't have much luck with image description. Thanks to advent of Encoder-Decoder architecture through DNNs, we have a way to compress the image to relevant important features, and these features were used as an input to NLP model (RNN) to generate the description of the image. In this paper, we explored Encoder using ResNet and Decoder using LSTM and trained our model on MSCOCO dataset and slight variation on design and hyperparameters,

2 Training

2.1 Data

We used the Microsoft COCO dataset and considered specific image ids for training and testing. As the original dataset is quite large, we used about $\frac{1}{5}$ th of it. Our training set contained around 16k images while the test set has around 3k images. The annotations are JSON files which contains image ids, image name, caption ids, captions etc. Most images contain multiple objects and significant contextual information, and each image comes with 5 human-annotated captions. The images create a challenging test bed for image captioning and are widely used in recent automatic image captioning work.

2.2 Network Design

The architecture for baseline LSTM model is same as the one given in assignment. RNN basedmodel simply uses an RNN instead of the LSTM in the architecture.

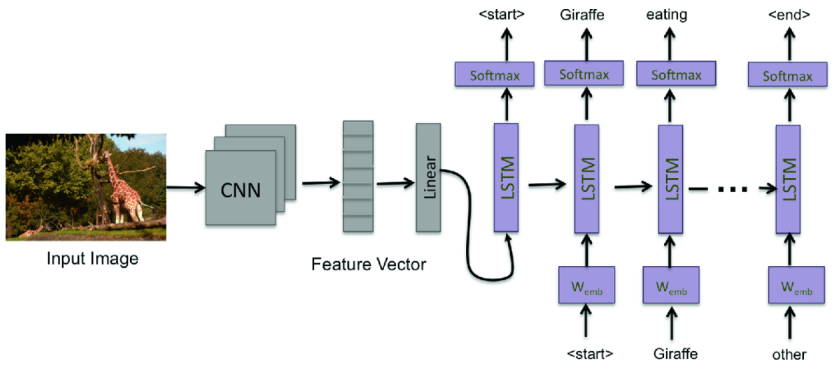

Figure 1: Model Architecture

The model architecture hyperparameters are given below and they are same for both LSTM and RNN based model so that we can make fair comparisons among the two methods.

Architecture
Since each image is of different sizes and there is fully connected layer in the encoder which we are fine tuning, we have to somehow give images of consistent sizes as input to the encoder. To overcome this, images are resized such that shorter dimension is scaled to 256 keeping the aspect ratio same. Resized images are then centre cropped to 224x224 image with the assumption that there is very little information along the edges of an image. The encoder takes this cropped image and gives out a feature vector of size 300 which is fed to the LSTM/RNN as input before training captions. Outputs from decoder are then reshaped to vectors of size 'vocab_size' using a linear layer. During training, these reshaped outputs are used for Cross-Entropy loss calculation using target captions.
Sampling Outputs
During testing and caption generation, for deterministic setting, the index corresponding to max value in softmax distribution is used to print out the caption token using an index-to-word dictionary. Under stochastic setting, for a given temperature, an index is sampled from the corresponding softmax distribution which is used for getting caption token.
Choosing Word Embeddings
In initial settings, we learn the word embeddings. Later, we experiment with pretrained Glove word embeddings, which is discussed in detail later in section **3.4**
Hyperparameters: Architecture hyperparameters are reported in Table **1**
LSTM Baseline: LSTM baseline is discussed in section **3.1**
Comparison with vanilla RNN: Discussed in section **3.2**

Table 1: Architecture Hyperparameters

crop size	batch size	embedding size	LSTM/RNN hidden size	LSTM/RNN input size	learning rate
224x224	64	300	750	300	1e-4

3 Discussion

3.1 Performance Discussion

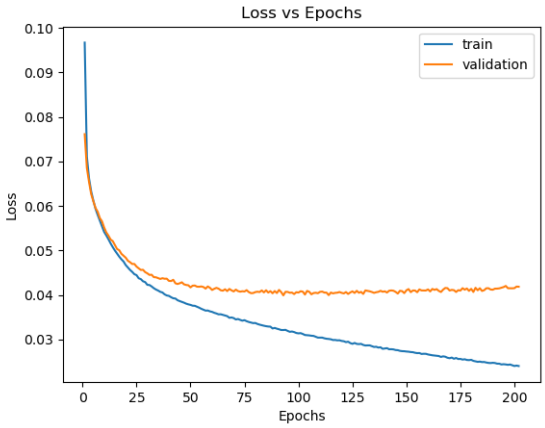

Figure 2: Loss vs Epoch for Baseline LSTM model

Table 2: Cross Entropy Loss and Perplexity Scores

Model	**Cross Entropy Loss**	**Perplexity Score**
RNN	0.041	1.042
LSTM Baseline	0.039	1.039
LSTM with Pretrained Embeddings	0.037	1.0377 height

Table 3: BLEU Scores

Model	**BLEU-1**	**BLEU-4**
RNN	58.94	7.48
LSTM with Embeddings	58.44	7.48
LSTM Baseline	57.06	7.17
LSTM with Stochastic sampling with t = 0.05	58.89	7.45
LSTM with Stochastic sampling with t = 0.1	58.99	7.41
LSTM with Stochastic sampling with t = 0.2	58.49	7.00
LSTM with Stochastic sampling with t = 0.5	55.54	5.61
LSTM with Stochastic sampling with t = 1	40.7	2.21
LSTM with Stochastic sampling with t = 1.5	15.47	0.83
LSTM with Stochastic sampling with t = 2	5.10	0.26

3.2 Comparison with Vanilla RNN

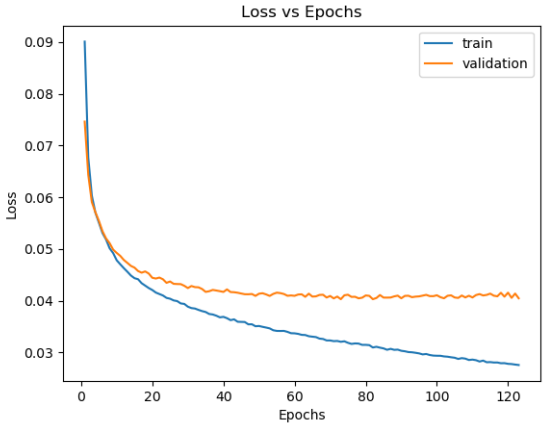

Figure 3: Loss vs Epoch for RNN model

Since LSTM has more parameters and bit more complicate than RNN, although it can achieve better performance, We expect that LSTM will take more time to learn than RNN. Thus, curves in the case of RNN will be more steeper and take less time to achieve satisfactory performance compared to LSTM

But the plots do not explicitly show the same, through our values, we saw that results are according to our expectations. For our chosen set of hyperparameters, RNN is learning only very very slightly faster and accordingly needs slightly lesser epochs to learn. At this point, we noticed that, our sentences are very small in length and hence there is not much difference in terms of model complexity. Also, it might be the case that vanishing gradient problem will lead to slower learning in RNN, thus making our comparison for RNN vs LSTM in terms of learning speed incomparable.

Table 4: Test losses and perplexity score

Model	**CrossEntropyLoss**	**PerplexityScore**
LSTM Baseline	0.041	1.042
RNN	0.037	1.0377

However, BLEU scores for LSTM is better than RNN which shows that LSTM is able to learn better language modelling. This gets reflected in the generated captions where we can see that RNN generated captions are not as good as LSTM ones.

We see that both RNN and LSTM perform comparably wrt loss and perplexity score. However, LSTM performs better as it is more complex. LSTM units include a 'memory cell' that can maintain information in memory for long periods of time. A set of gates is used to control when information enters the memory, when it's output, and when it's forgotten. This architecture lets them learn longer-term dependencies. This helps LSTMs to learn better than RNNs. The results also confirm the same.

Note that, we see both LSTM and RNN have almost similar performance. Looking at the training captions, we observed that most captions are lower than 20 words. Since LSTM do not suffer from vanishing gradient problems, the difference in performances between LSTM and RNN become more apparent as sequence length increases. Since we are considering overall average in our experiment, we feel the two models have similar performances.

3.3 BLEU score comparison

We can observe that LSTM performs best among all the models in terms of BLEU scores. We expect BLEU score is bit different compared to cross entropy loss. While cross entropy loss is continuous function, BLEU score is dependent on sentences and output words, thus chage in BLEU score is not a gradual process in terms words, and thus there is not an exact correlation with loss in finer details. Because of this, it can happen that BLEU score is better, but cross entropy loss is worse for some model. For example in our case, this happened in the case of comparing baseline model and model with temperature = 0.1. At the same, in larger scale better model, will have better BLEU score. In our case, we found that LSTM / LSTM with temperature 0.1 are better than others and have better BLEU scores.

Table 5: BLEU Scores

Model	**BLEU-1**	**BLEU-4**
LSTM Baseline	58.94	7.48
LSTM with Embeddings	58.44	7.48
RNN	57.06	7.17
LSTM with Stochastic sampling with t = 0.05	58.89	7.45
LSTM with Stochastic sampling with t = 0.1	58.99	7.41
LSTM with Stochastic sampling with t = 0.2	58.49	7.00
LSTM with Stochastic sampling with t = 0.5	55.54	5.61
LSTM with Stochastic sampling with t = 1	40.7	2.21
LSTM with Stochastic sampling with t = 1.5	15.47	0.83
LSTM with Stochastic sampling with t = 2	5.10	0.26

3.4 Exploration with Stochastic approach

Our results are in the **table 3**. Please refer to them

we were expecting when temperature is very small, our outputs will be similar to thecase of determenistic outputs and when temperature is large, outputs will be random as

$$\lim_{t \to 0} \text{distribution} = \begin{cases} 1 & \text{for max value} \\ 0 & otherwise \end{cases}$$

$$\lim_{t \to \infty} \text{distribution} = \frac{1}{\text{num classes}} = \text{uniform distribution}$$

Thus, when t is near zero, we are predicting almost similar to the deterministic case, hence performance should be near to that. As t grows to larger value, difference between probabilities becomes smaller and smaller and thus, our sample grows more and more towards uniform distribution.

Our results showed the same, thus reinforcing the same ideas. We also observed that determistic model can be corrected with temperature value as data can have noise or may not representative of the real vocabulary. Thus, for smaller values of temperature (depends on data (size and context), in our case it is [0.05, 0.2]), our model becomes more robust performs better on test data.

3.5 Experiments with Pretrained Embeddings

We chose pretrained GloVe embedding because it handles both global and local language semantics compared to Word2Vector. Observing Table 2, we see that LSTM with pretrained GloVe embeddings perform better than learning embedding layer during training in terms of loss function. Note that Table 3 BLEU scores for both models are similar, but thats only because BLUE scores are not a direct representative of losses. We think the reason behind GloVe embeddings being better is that since our corpus is limited, the layer embedding trained model would be more biased towards vocab within the data. If there were an abundance of corpus data, the two methods should yield similar losses.

We also expect that if smaller vocabulary size leads to high bias and thus addding pretrained embedding will result in better model where as when we have abundant vocabulary, pretrained will not have much effect, if anything it will add just variance to the model. We also saw this with training different vocabulary sizes

Table 6: BLEU Scores

Model	**BLEU-1**	**BLEU-4**
LSTM with Pretrained Embeddings	58.44	7.48

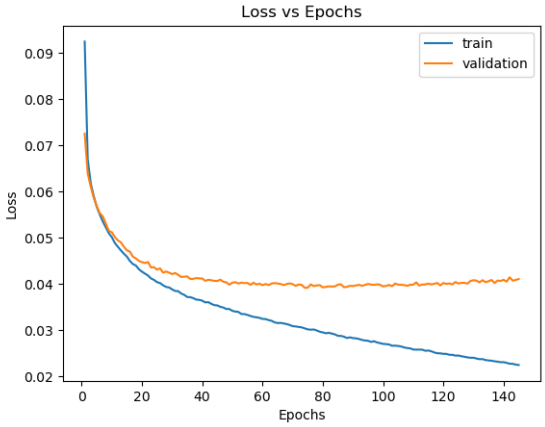

Figure 4: Loss vs Epoch for pretrained Glove word embedding

Comparing above figure with the training loss curves for layer embedding process, we see that pretrained word embeddings learn much faster. Layer embedding model takes around 60 epochs to get the best validation loss whereas it takes only 40 epochs for GloVe embedding to reach the same loss value. This makes sense because the network does not have to spend time finding the word to vector mappings and can directly optimize the LSTM part.

4 Results display

Figure 5: Surfing Figure

Table 7: Captions for sufer image

Model	**Caption**
Real	A surferboarder is surfing on a small wave
LSTM Baseline	a man in a wetsuit riding a wave on a surfboard
LSTM with Embeddings	a man riding a surfboard on top of a wave
RNN	a man riding a surfboard on a wave in the ocean
LSTM with Stochastic sampling with t = 0.05	a man in a wetsuit surfs on a surfboard
LSTM with Stochastic sampling with t = 0.1	a man riding a wave on a surfboard in the ocean
LSTM with Stochastic sampling with t = 0.2	a man riding a wave on a surfboard
LSTM with Stochastic sampling with t = 0.5	a man riding a surfboard on a wave
LSTM with Stochastic sampling with t = 1	motorcyclist in a large large wave swimming pool
LSTM with Stochastic sampling with t = 1.5	an parasailing chicago glazed cute if orange

Figure 6: People Standing

Table 8: Captions for image above

Model	**Caption**
Real	Two people sitting at a table having a conversation
LSTM Baseline	a group of people standing in a room with a wii
LSTM with Embeddings	a group of people standing around a table with a video game
RNN	a group of people standing in a living room
LSTM with Stochastic sampling with t = 0.05	a group of people standing in a room with a wii controller
LSTM with Stochastic sampling with t = 0.1	a group of people standing around a table with a wii remote
LSTM with Stochastic sampling with t = 0.2	a group of people standing in a room with a man
LSTM with Stochastic sampling with t = 0.5	a group of men standing next to each other in a living room
LSTM with Stochastic sampling with t = 1	a group of people sitting and women stand next to each other
LSTM with Stochastic sampling with t = 1.5	the pose zombie shelter la doorways to juggling friends in formal hands

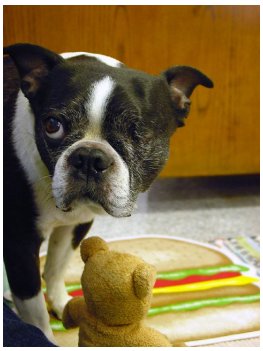

Figure 7: Dog Image

Table 9: Model Caption

Model	**Caption**
Real	A dog standing in a kitchen with a toy
LSTM Baseline	a small dog is sitting on a couch with a dog
LSTM with Embeddings	a dog is sitting on a chair with a dog
RNN	a dog is playing with a dog on a leash
LSTM with Stochastic sampling with t = 0.05	a small dog is sitting on a couch with a dog
LSTM with Stochastic sampling with t = 0.1	a small dog and a dog on a chair
LSTM with Stochastic sampling with t = 0.2	a small brown and white dog playing with a dog
LSTM with Stochastic sampling with t = 0.5	a small dog is standing on a chair ¡unk¿
LSTM with Stochastic sampling with t = 1	a small small dog holds up on a windowsill ¡unk¿
LSTM with Stochastic sampling with t = 1.5	a counter makes an full excited great kitty ¡unk¿

We observed that when trained on whole dataset instead of the subset (through ids given to us), we are getting a huge improvement (BLEU Score 80, 30). But here, we are mentioning only results according to the data set mentioned in the assignment.

5 Conclusion

At the start, It was unclear and very sruprising to see the results of CNN-LSTM architecture work in lecture slides. But seeing how good the the captions generated by the model both in terms of grammar and description, we are really inspired and got to learn a lot of new things, reinforcing some great ideas of LSTM.

6 Author's contributions

Everyone contributed equally towards coding, discussion, debugging, training and monitoring models.

6.1 Astuti Sharma

Dataloader, Dataset analysis, perplexity scores

6.2 Divyanshu Mund

Stochastic approach (using Temperature), BLeU scores, output generation

6.3 Kunal Jain

LSTM, RNN models, Expermenting with Embeddings

6.4 Ria Aggarwal

LSTM model, report, tuning hyperparameters

6.5 Sai Akhil Suggu

RNN model, comparing different models, experimenting with embeddings