Neural Image Captioning

Pulkit Aggarwal, Sidharth Gulati, Ishan Upadhyaya {AGGARWALPULKIT23, SIDHARTH.GULATI, ISHAN793}@GMAIL.COM
Electrical Engineering Department, University of California, Los Angeles

1. Introduction

Automating Image captioning presents an interesting challenge, combining ideas from the field of computer vision and natural language processing. The task is significantly harder than classical vision tasks such as Object detection or image classification due to the fact that captioning an image needs to capture objects in an image *and* generate a "story" describing the relation between them. Moreover, for a *good* "story", captioning must be done in a natural language like English, requiring strong language models.

In this project we explore the approach similar to one proposed by (Vinyals et al., 2014) where we first "encode" an image and then "decode" it with a Recurrent Neural Network(RNN). We look at the two following two ways to encode an image:

- **Encoding using a Linear Layer -** In this approach, we encode the image using a linear neural network layer. The output from this linear layer is then used to initialize a Long Short Term Memory (LSTM) cell for generating the target caption.
- **Encoding using Convolutional Neural Network (CNN) -** In this approach, we use CNN to encode an image as they can extract complex feature maps that successfully capture the essence of an image. The encoded image is then used to initialize a Recurrent Neural Network(RNN) for generating captions.

Further, we explore two different types of RNN listed below:

- **Gated Recurrent Unit (GRU)** - Here we use a GRU cell to generate caption as they have less parameters to train and hence, require less training time.
- **Long Short Term Memory (LSTM)** - Here we use a LSTM cell to generate caption as they have a internal memory unlike GRU and have shown better empirical results in sequencing task.

Our report is structured as follows. In section 2, we explain the caption generating model with a brief description of LSTM, GRU and Convolutional Neural Network. Then, in section 3 we describe the experiments we carried out and the corresponding results. Finally, in section 4 we conclude by comparing our hypotheses with results from our experiments.

2. Caption Generator Model

In this project we look at Image captioning as a sequence generating task conditioned over a given image. Recent research has shown that Recurrent Neural Networks(RNN) produce State-of-Art results for similar tasks, such as Machine translation, which we can be thought of as Sequence generation conditioned over a given input sentence. Part of the appeal of using RNN is their ability to optimize the probability of correct description in an end to end fashion. With this framework in mind, we look at the following optimization problem as proposed in (Vinyals et al., 2014)

$$\theta^* = \underset{\theta}{\text{argmax}} \sum_{I,S} \log p(S|I) \tag{1}$$

Here θ represents the model parameters, *I* is an image and *S* is the corresponding sentence. For a sentence of length N, the conditional probability can be written as:

$$\log p(S|I) = \sum_{t=0}^{N} \log p(S_t|I, S_0, S_1, \ldots, S_{t-1}) \tag{2}$$

A single RNN cell represents a natural way to model the summand in equation (2), as the state of sentence up to time step $t-1$ can be encoded in the hidden state h_{t-1}. Probability of the current state and the new hidden state of the network can be computed using non-linear functions f and g from past information as follows:

$$\begin{aligned} h_t &= f(x_t, h_{t-1}) \\ p_t &= g(h_t, x_t) \end{aligned} \tag{3}$$

To initialize the network, we use an image encoded with a Linear Layer or Convolutional Neural Network as shown in Figure (1). During training, we solve the problem presented in (1), treating *(S,I)* as a training example pair and optimize the sum of the log probabilities given in (2) over the whole training set using stochastic gradient descent (as discussed

in Section 3). In the following two subsections, we briefly describe the two RNN model we've used.

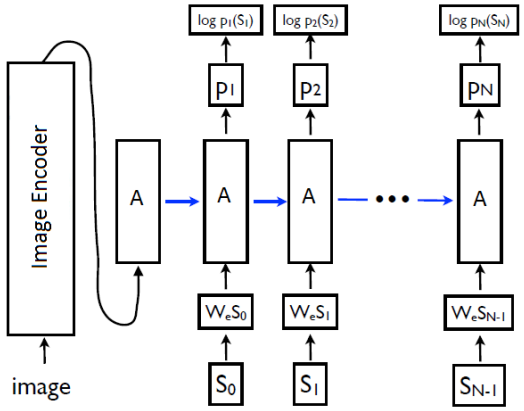


Figure 1. End-to-End model.

$$
\begin{aligned}
f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f) \\
i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i) \\
\tilde{C}_t &= \tanh(W_C[h_{t-1}, x_t] + b_c) \\
C_t &= C_{t-1} * f_t + i_t * \tilde{C}_t \text{ // Multiplicative} \\
o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\
h_t &= o_t * tanh(C_t) \text{ // Multiplicative} \\
p_t &= \text{Softmax}(h_t)
\end{aligned} \tag{4}
$$

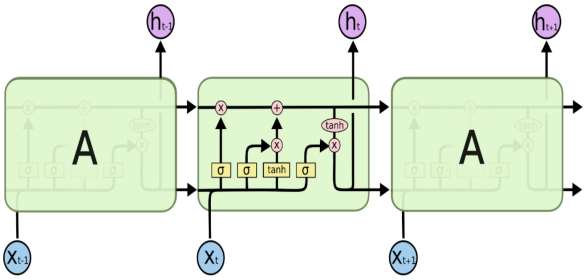


Figure 2. The repeating module in an LSTM contains four interacting layers.

2.1. Recurrent Neural Network

2.1.1. LSTM

The Strength of any RNN is primarily governed by the choice of the functions f and g as they decide the expressiveness of each cell and it's ability to handle vanishing and exploding gradients. LSTM introduced in (Hochreiter & Schmidhuber, 1997) makes a *"good"* choice of such functions through a interesting architecture and the idea of multiplicative interaction, as shown in 2.

A large part of the success of LSTM's is through it's "memory state" running through all the cells, encoding the past observation. Every cell part of the network interacts with this state through "*gates*", which are linear layers applied multiplicatively to the memory state. The multiplicative nature of such gates allows the cells to control the extent to which:

- A value in the current state is "forgotten" via the forget gate, f
- Input Values are "read" through the input gate, i
- Impact memory, input and past hidden state have on the output and/or the current hidden state. o

Mathematically, we represent these ideas as follows:

Such multiplicative operations allow the LSTM cell to efficiently handle vanishing and exploding gradients encountered while learning the matrices W and the biases b. We also apply a Softmax layer over the outputs of the gates to generate a probability distribution over the words.

2.1.2. GRU

Inspired by LSTM, (Cho et al., 2014) introduced a different hidden unit known as GRU, also making use of multiplicative weights with gates to update the hidden state of the network. However, unlike a LSTM cell, a GRU based network does not maintain a memory state and uses the information provided by the current input to alter the state of the network. The update equations for this cell are:

$$
\begin{aligned}
z_t &= \sigma(W_z[h_{t-1}, x_t]) \\
r_t &= \sigma(W_r[h_{t-1}, x_t]) \\
\tilde{h}_t &= \tanh(W[r_t * h_{t-1}, x_t]) \\
h_t &= (1 - z_t) * h_t + z_t * \tilde{h}_t \\
p_t &= \text{Softmax}(h_t)
\end{aligned} \tag{5}
$$

Based on the update equations, we can make the following observations

- The reset gate(r_t) decides whether or not to reset the hidden state of the network.
- The update gate(z_t) controls the amount of information that flows from the previous state to the current state.

Empirical studies reported in (Cho et al., 2014) demonstrate that a GRU based RNN network performs comparably to a network using LSTM's while using lower number variables.

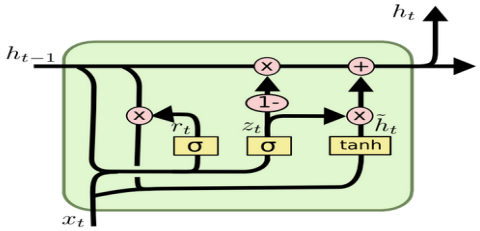


Figure 3. A Gated Recurrent Unit model. Note that unlike a LSTM, GRU doesn't have a memory state

2.2. Convolutional Neural Network (CNN)

2.2.1. Motivation

As captioning an image requires informative features to explain the images in short and/or long sentences, we felt Deep neural network would be a suitable encoder as they are efficient in capturing semantic information embedded within images. Moreover, in recent times, deep neural networks, such as CNN have outperformed shallow machine learning models in classification tasks such as ImageNet(Russakovsky et al., 2015) etc. This is because the higher layers in a deep model are able to reuse primitives obtained from the lower layers and learn more complex features maps. Moreover, CNN is invariant to the image position as we look at each receptive field of the image and extract features locally preserving the relative position of the learned features. Due to these reasons, we decided to experiment with Convolutional Neural Network to encode an image and use them as an initial state of LSTM structure.

2.2.2. Inception

We used Inception V3 architecture (Szegedy et al., 2015) as a Convolutional Neural Network. Inception v3 has reported best results so far, a top-5 error of 6.1% and a top-1 error of 22% in the ImageNet challenge. Moreover, Inception architecture has shown great performance in terms of computational cost which makes it suitable for big data scenarios where we need to process data at a reasonable cost. Hence, given these advantages, we implemented our model using transfer learning.

3. Experiments

3.1. Training Details

In this project we used a CNN model with the inception-v3 architecture initialized from the weights learned by training on the ImageNet dataset (Russakovsky et al., 2015). These weights can be fine-tuned on MSCOCO dataset or they can be used without any change. However, fine-tuning the inception-v3 model is a slow process, taking considerable amount of time and as discussed in (Vinyals et al., 2014), it does not help much in improving the results. Thus, we directly used the pre-trained weights without any change.

Next step includes learning a LSTM layer. As discussed in section 2, the linear layer from the image encoder to the RNN(image embedding) and the input sequence(sequence embedding) to the LSTM cells are learned simultaneously with the weights of LSTM model. To study the effect of the embedding vector size, we tried various sizes $\{512, 256, 128\}$ to generate the embedding vector.
We trained all the weights using Stochastic gradient descent with a batch-size of 32, a fixed learning rate and no momentum. In order to handle the problem of over-fitting, we used the pre-trained weights of Inception-v3 architecture and dropout with a keep_probability of 0.7.

3.2. Dataset

For our experiments, we used the MSCOCO dataset (Lin et al., 2014). It is the largest dataset available for the task of image captioning and has 82783 training images, 40504 validation images and 40775 testing images(captions are not available). Each image in mscoco train/val set has been annotated with 5 sentences. These image-caption pairs are used as a separate training instance in the training data. The entire training set was converted into 256 shards with each shard containing 2300 image-caption pairs.This gave us 586363 image-caption example per epoch.
For reporting the BLEU-4 score(Papineni et al., 2002) results, we have randomly sampled 4K images from the validation set and used them as test set.

3.3. Results

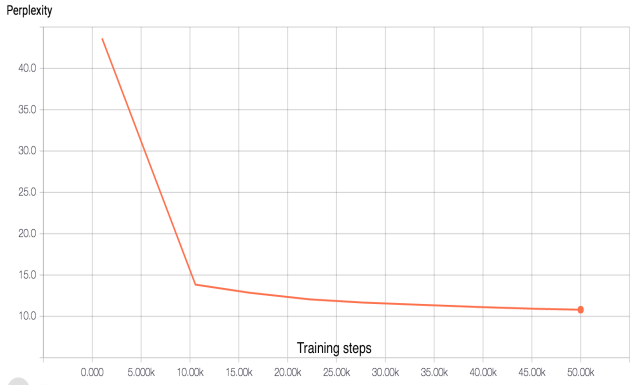
While training, we looked at the perplexity values[1] on the validation set and the cross-entropy loss of the model. We compared the performance of different models using BLEU score instead of perplexity as the former is preferred in various research papers. Figure 4 compares perplexity for GRU-512 4(a) and LSTM-256 [2] (4(b)).

The total_loss curve was nearly the same for all models. Figure 5 shows the curve for LSTM-256 model.

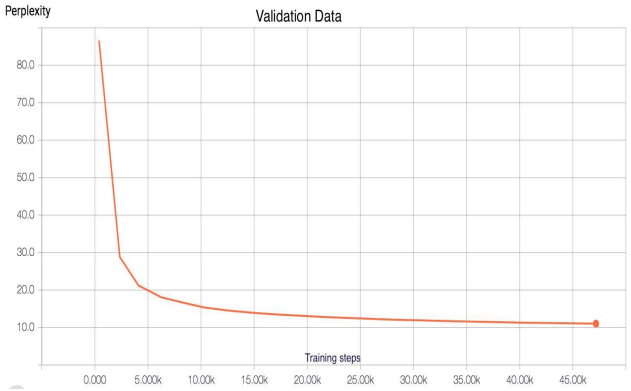
The embedding used in LSTM model have an advantage that they are independent of the vocabulary size. Also, we focused on using a smaller size of the embedding vector which can give results comparable to larger sizes.
Table 1 shows the captions generated on some images for 3 different embedding sizes. It is clearly visible that captions

[1]The perplexity is the geometric mean of the inverse probability for each predicted word

[2]LSTM/GRU-x indicates the model trained with LSTM/GRU cell and length of embedding vector is x.



(a) GRU-512



(b) LSTM-256

Figure 4. Perplexity for GRU-512 and LSTM-256.

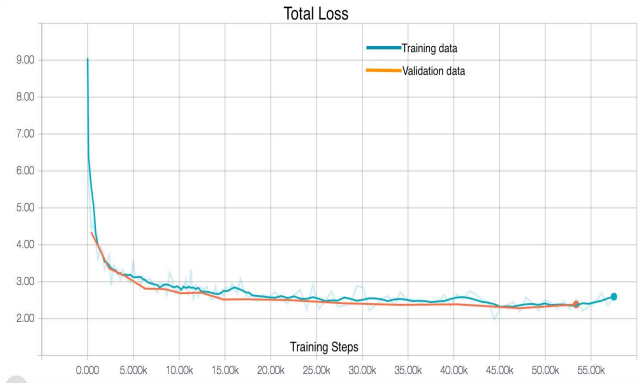


Figure 5. Training Loss

generated from size-512 are better.

Further, we observed that for "simple images" (having only one object and a solid background (A.3)), the captions generated are nearly the same for all the 3 sizes. This may be because the features of "simple images" are captured equally by the 3 embedding sizes.

In our model, the CNN component extracts the features of the images and then those features are converted to image_embedding using a linear layer. We wanted to compare the performance of a model with CNN and a model

(a) Fig A (b) Fig B

(c) Fig C (d) Fig D

Figure 6. Images to show difference between captions for different embedding sizes (See Table 1 for the captions.)

Table 1. Difference between captions generated by embedding vectors of different sizes

Figure 6	Embedding Size(a=128,b=256,c=512)
6(a)	a) a couple of giraffe standing next to each other. b) giraffe standing next to a wooden fence. c) group of giraffe standing next to each other in a cage.
6(b)	a) a young boy is brushing his teeth with a toothbrush. b) a young boy with a toothbrush in his mouth. c) a group of young boys sitting on top of a bed.
6(c)	a) a man is riding a horse on a beach. b) a dog with a frisbee in its mouth. c) a dog jumping in the air to catch a frisbee.
6(d)	a) there is a plate of food on the table b) a plate of food that is on a table. c) a table topped with plates and bowls of food.

without CNN. We trained a No-CNN model, with only a linear layer used to create the image_embedding vector. As expected, this linear layer was not powerful enough to create distinctive enough features from the images. The generated captions for the No-CNN model were unrelated to the image (therefore,we have not computed the BLEU-4 score for this model). But this No-CNN experiment has shown us the importance of CNN models in image captioning and how these pre-trained CNN architectures are able to help in feature extraction for our task.

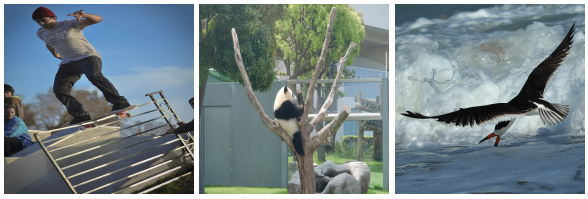
For the BLEU-4 score of a generated caption, we use all the 5 available reference captions for that image. There are other metrics available for evaluating the generated captions(Vedantam et al., 2014), but in recent works BLEU-4 score is the standard metric. BLEU-4 score was computed

on COCO-4K images as discussed in 3.2. Table 2 shows the score on different configurations.

Table 2. BLEU-4 score on different configurations

Configuration	BLEU-4 score
LSTM-128	58.469
LSTM-256	58.751
GRU-512	58.988
LSTM-512	59.365

Below are some results, from Figure (7) to Figure (9) for the LSTM-512 model. These images show the performance of our model in decreasing order (from "captions without errors" to "captions unrelated to the image").



(a) man riding a skateboard down the side of a rail. (b) Panda bear sitting on a tree branch. (c) Bird flying over a body of water

Figure 7. Captions Without Errors

(a) a woman walking down street holding an umbrella. (b) a train traveling a down tracks next to a forest. (c) a woman holding an umbrella on a beach.

Figure 8. Captions With Minor Errors

(a) a woman holding a toothbrush in her mouth. (b) a young boy swinging a baseball bat at a ball. (c) a man riding a skateboard down the side of a ramp.

Figure 9. Captions Unrelated to the Image

Based on the experiments, we observed that GRU-512 model generated some captions which are better than it's LSTM counterpart. For e.g, in the first row of table 3 we observe that GRU was able to distinguish that this is not a 'stop' sign and it has white color also, something the LSTM didn't capture so well. On the other hand, we observed that LSTMs are better in capturing long-term dependencies than GRU, as can be seen in the second row of table 3.

Caption	Image
GRU:a red and white sign in front of a building. **LSTM**:a red stop sign sitting on the side of a building.	
GRU:a person jumping a skate board in the air **LSTM**:a man riding a skateboard down the side of a rail.	

Table 3. Comparing the results of GRU and LSTM networks.

4. Conclusion

In this project we explored the task of Image captioning. For this, we performed experiments which were based on (Vinyals et al., 2014) and studied the performance for different embedding vector sizes. Moreover, we also studied the impact of different RNN structures, namely LSTM and GRU. Based on our experiments we found that for "simple images" the results are nearly same for all embedding vector sizes, while 512 vector size generates comparatively more descriptive captions. It is evident from the experiments, that LSTM-512 performed the best with the highest BLEU-4 score of 59.365. GRU cell also performed nearly the same as that of LSTM, however, it was unable to capture long-term dependencies in captions. Moreover, to study the importance of the features extracted from CNN, we trained a No-CNN model which generated unrelated captions. Since, we are training the image embedding simultaneously in No-CNN model (using a linear layer), we hypothesize that a pre-trained model on a similar task will perform better than the current No-CNN model. As extension to the work we did in this project, we would like explore the above stated hypothesis. Furthermore, it would also be interesting to see how different feature extraction methods like Stacked Denoising Autoencoder affect the performance of such a system. Lastly, we can explore the effects of adding an Attention Layer (as discussed in (Xu et al., 2015)) to the LSTM cell for generating image captions or using a character level LSTM.

Acknowledgments

We greatly acknowledge Professor Raghu Meka for his valuable discussions and suggestions on the project which helped us to substantially improve the quality of our work.

References

Cho, Kyunghyun, Van Merriënboer, Bart, Gulcehre, Caglar, Bahdanau, Dzmitry, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Lin, Tsung-Yi, Maire, Michael, Belongie, Serge J., Bourdev, Lubomir D., Girshick, Ross B., Hays, James, Perona, Pietro, Ramanan, Deva, Dollár, Piotr, and Zitnick, C. Lawrence. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL http://arxiv.org/abs/1405.0312.

Papineni, Kishore, Roukos, Salim, Ward, Todd, and Zhu, Wei-Jing. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 311–318. Association for Computational Linguistics, 2002.

Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, Bernstein, Michael, Berg, Alexander C., and Fei-Fei, Li. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Shallue, Chris. Show and tell: A neural image caption generator. 2016. URL https://github.com/tensorflow/models/tree/master/im2txt/im2txt.

Szegedy, Christian, Vanhoucke, Vincent, Ioffe, Sergey, Shlens, Jonathon, and Wojna, Zbigniew. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015. URL http://arxiv.org/abs/1512.00567.

Vedantam, Ramakrishna, Zitnick, C. Lawrence, and Parikh, Devi. Cider: Consensus-based image description evaluation. *CoRR*, abs/1411.5726, 2014. URL http://arxiv.org/abs/1411.5726.

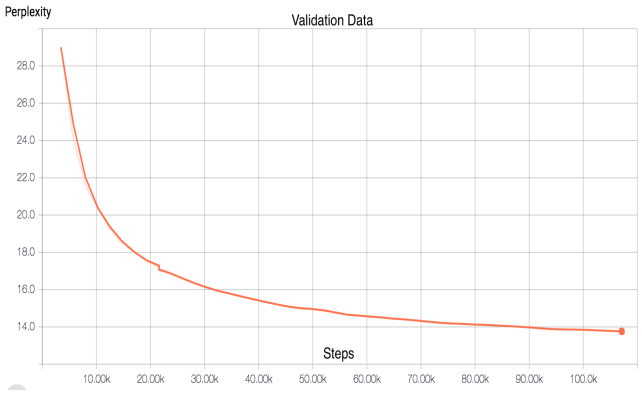
Vinyals, Oriol, Toshev, Alexander, Bengio, Samy, and Erhan, Dumitru. Show and tell: A neural image caption generator. *CoRR*, abs/1411.4555, 2014. URL http://arxiv.org/abs/1411.4555.

Xu, Kelvin, Ba, Jimmy, Kiros, Ryan, Cho, Kyunghyun, Courville, Aaron, Salakhutdinov, Ruslan, Zemel, Richard S, and Bengio, Yoshua. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2(3):5, 2015.

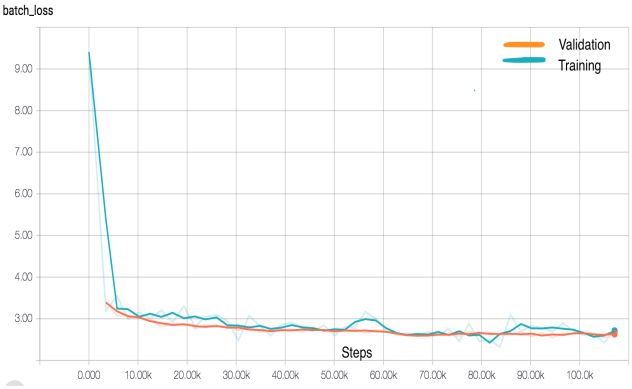
A. Appendix

A.1. Evaluation/Training Plots

Figure (10) shows the perplexity and batch loss for No-CNN model.



(a) Perplexity

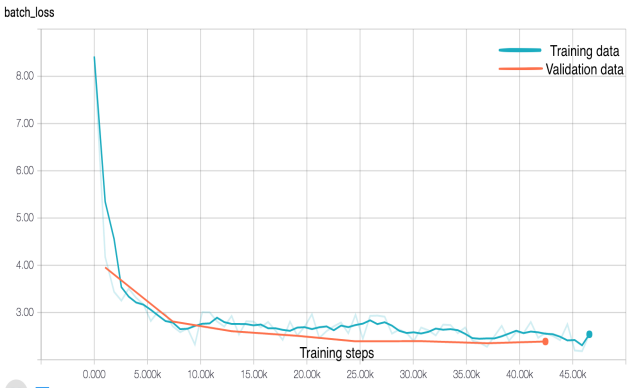


(b) Loss

Figure 10. Perplexity and Batch Loss for No-CNN.

Figure (11) shows the perplexity and batch loss for LSTM-128 model.

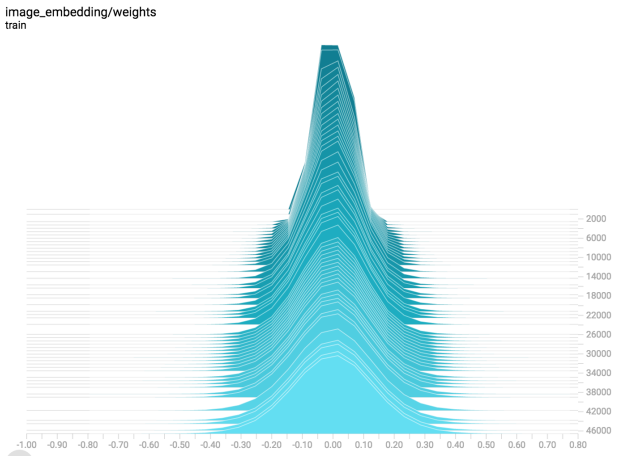
(a) Perplexity



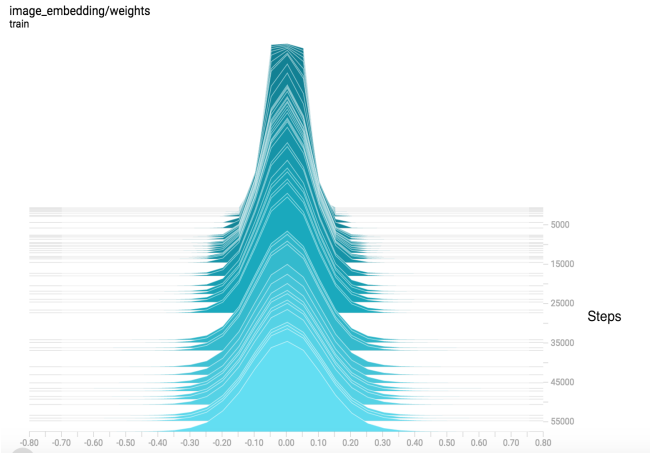
(b) Batch Loss

Figure 11. Perplexity and Batch Loss for LSTM-128

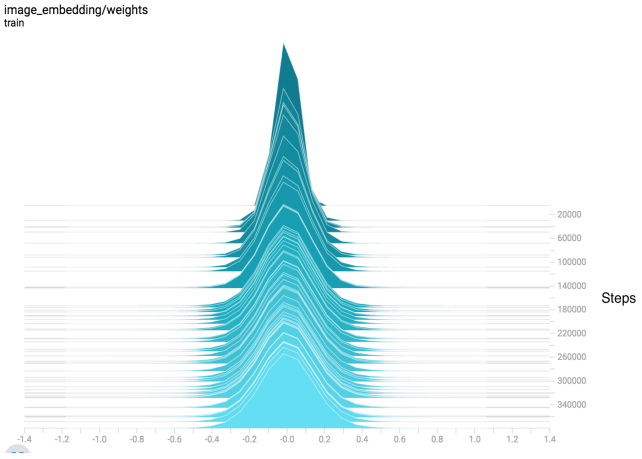
Figure (12) shows the Image Embeddings distribution for LSTM-128 (12(a)), LSTM-256 (12(b)) and LSTM-512 (12(c)) models respectively.



(a) LSTM-128



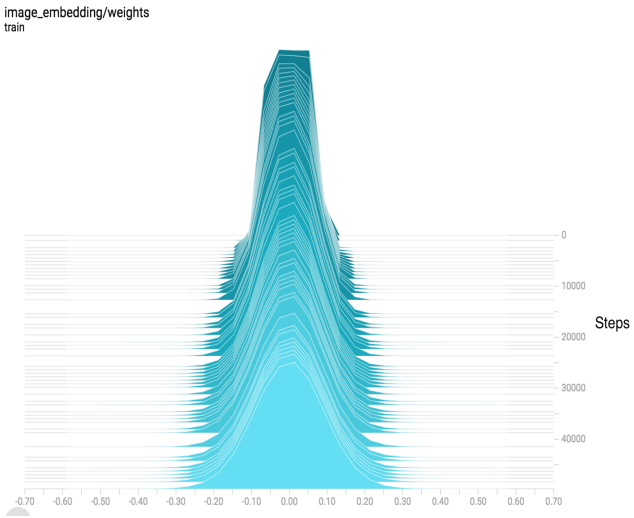
(b) LSTM-256



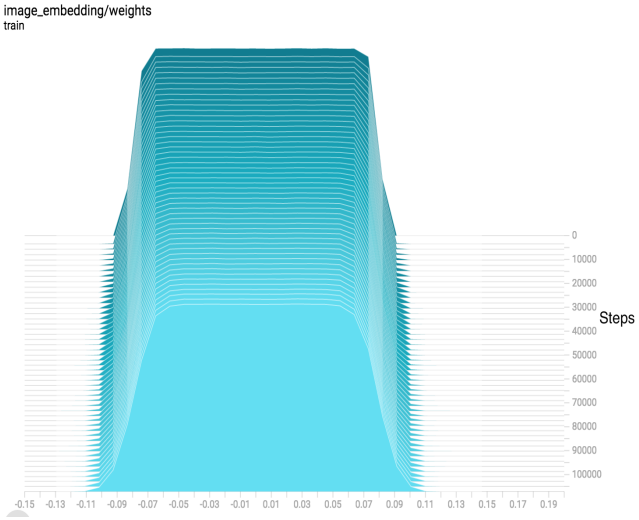
(c) LSTM-512

Figure 12. Image Embedding distribution for LSTM-128, LSTM-256 and LSTM-512.

Figure (13) shows the Image Embedding distribution for GRU-512 (13(a)) and No-CNN (13(b)) models respectively.



(a) GRU-512



(b) NO-CNN

Figure 13. Image Embedding distribution for GRU-512 and No-CNN.

A.2. No-CNN Model Image Captioning Results

Below are some results generated by our No-CNN model, for the images in Figure (14), where the LSTM-512 model was generating captions without errors.

(a) A (b) B (c) C

Figure 14. Images from Figure 7

The caption generated by No-CNN model for Figure (14) is given below:
14(a) man sitting on a bench in a park.
14(b) man in a white shirt and black shorts playing a game of tennis.
14(c) man riding a wave on top of a surfboard.
Seeing these results, we can say that No-CNN model is not able to generate captions related to the image.

A.3. Simple Images

(a) A (b) B (c) C

Figure 15. Simple Images

These are ”simple images” which have only one object and solid background. For ”simple images” as discussed in section 3.3, the captions generated by LSTM-x, where x $\in$ {128,256,512} are nearly the same.

A.4. Implementation

Our implementation of the project in tensorflow is motivated from (Shallue, 2016).