Image Captioning with Visual Attention

Pankaj Bhambhani, Natcha Simsiri, Daniel Sam Pete Thiyagu
College of Information and Computer Sciences
University of Massachusetts Amherst
{pbhambhani, nsimsiri, dthiyagu}@cs.umass.edu

***Abstract*— Using gradient descent based training algorithms incorporated with an attention mechanism, we were able to build a deep learning model that learns to generate image descriptions given an unseen image. Inspired by past work in treating the caption generation task as a neural translation task, our deep learning network tries to represent natural language, visual features and visual focal points in the same embedding while using a recurrent network to generate sentences. We implemented and evaluated our attention model with the traditional neural approach to image captioning as well as non-deep learning methods. Our evaluations were done using the bleu, meteor and cider metrics. The results show that attention-based deep learning models can learn to produce good image captions. We also show visualizations to infer how a model is able to focus on certain parts of the image while predicting the next word.**

I. INTRODUCTION

Image captioning is the task of describing an arbitrary image in a syntactically-proper sentence in the human language (for our case would be English). Training a computer to be able to automatically do this process is a very challenging task that requires both insights from the field of natural language processing and computer vision. Object identification techniques from computer vision has been studied very well over the past years - however, image captioning is a much more complex task. Image captioning must not only identify objects contained in an image, but it also must express how these objects relate to each other. To do so we must find a way to represent these features we want to learn in an embedding that captures the high level semantic of what was previously described. Furthermore, we must be able to convert this high level understanding of the image to a natural language like English, which means that a language model is needed in addition to visual understanding.

Our inspiration for solving the task of image captioning comes from the state of the art technique in machine translation. In machine translation, we have a sentence S from one language, and we want to translate this to a sentence T in another language by learning a model θ and maximize $p(T|S, \theta)$. Similarly, for our task we seek to learn a model θ and maximize $p(S|I, \theta)$ where S is the sentence we would like to generate and I is the image. In the translation case, the state of the art was achievable by representing the probability distribution via a Recurrent Neural Network (RNN) that provides a robust way to encode an arbitrary length sequential information into a fixed length vector representation. Furthermore, the RNN could also be used to decode the representation and produce back a sentence.

For image captioning we will leverage the "encoder-decoder" architecture used in the state of the art neural translation technique described previously. The challenge is finding a way to embedding inputted images, i.e representing visual semantics. Convolution Neural Networks (CNN) has shown to be a very promising deep learning method that learns image features and is able to represent them into a fixed length vector. Examples are the Oxford VGGNet or the GoogleNet (citation). We can use a trained CNN and "connect" the network to an RNN decoder to generate sentences. The in-depth details of the architecture will be described in the later sections

Incorporating visual attention in deep learning for doing tasks such as image captioning has grown in popularity in the recent years. This idea of visual attention is inspired from the human biology in how we generally do not perceive the world at once, but only portions of the things we see while 'graying' out the surrounding. Translating this idea to our architecture, instead of producing captions from a static image feature vector, we can caption from various portions of the image feature vector. In other words, for each time step we generate a new word, we would have a different image embedding to work with.

In this paper we will go in-depth of our various implementations of the neural image captioning system inspired by previous work. We will do a comparative study between each of the following techniques: Vanilla image captioning deep learning network that uses VGG16/GoogleNet as our image model and an LSTM/GRU/Soft-Attention LSTM as our language model. Furthermore we also ran experiments with non-deep learning methods, which includes SVM and Logistic Regression.

II. RELATED WORK

For this section we will outline other published work in image captioning - specifically looking at other deep learning models. In the recent years there has been a growing number of works that uses the "encoder-decoder" recurrent deep learning model in doing image captioning. This is because we can simply treat image captioning as a translation problem where instead we treat an image as just another "language" by representing images the same way as as we represent languages.

A. Multimodal Neural Language Models

This image captioning model follows the encoder-decoder approach where the model treats image captioning as a neural translation task such that as supposed to translating from one language to another language - the model translates "images" to some language. In other words, the model treats an image as just another language. The encoder-decoder is often described in terms of a Recurrent Neural Network - a type of neural learning network that is able to embed arbitrary length sequence of inputs as a fixed length vector. A deep convolutional neural network is used to build a feature vector for images and is projected to the embedding represented by the language model's "encoder" - or in this case a long-short term memory network - a type of recurrent neural network. This combined image-sentence embedding is referred to as a multimodal language model. When generating sentences, the paper introduces a Structured-Content Neural Language Model - an embedding in the "decoder" recurrent neural network that uses the multimodal embedding.

The key idea is to learn a multimodal representation given an image and a sentence during training. Suppose D is the dimension of the image's feature vector, and V is the vocabulary size, the image embedding matrix is denoted by W_i is a $K \times D$ matrix while the word embedding is a $K \times V$ matrix. An input sentence in the training data is $S = w_i, ..., w_N$ in space R^n for each word, but represented in the hidden layer of an LSTM in a vector v for each time step. The gradient descent training will try to optimize the following pairwise ranking loss function:

$$min \sum_x \sum_k max\{0, a - s(x, v) + s(x, v_k\} + \sum_v \sum_k max\{0, a - s(x, v) + s(x, v_k\}$$

Where the function s is a "score" that indicates similarity between the sentence embedding and image embedding and $x = W_I \times q$ where q is an image feature vector. Training on such loss function has shown to give good understanding of sentence "similarity" - for example the vectors of "images of blue cars" is similar to the vectors "images of red cars". The gist of the paper comes down to their language model - call the "Structured-Content Neural Language Model" where given the sentence S a multimodal embedding u and a word-specific structure variable - for example the part of speech, the goal is to model the following distribution:

$$P(w_n = i | w_1^{n-1}, t_n, u)$$

Which models the distribution of the next word given previous description, and the language structure t and the multimodal embedding u. Kiros et al. 2014

B. Deep Visual-Semantics Alignments

Building from similar concepts mentioned in the above section, the goal of this paper's technique is to not only learn the multimodal embedding between the image and sentence, but it introduces some sense of "closeness" for semantically similar objects. The deep learning network is organized similar to the previous paper - a variant of CNN is used to create an image embedding, while an RNN is used to represent the language model. However, instead of projecting a bag-of-words representation of the sentence to the same embedding space as the image embedding, the words embedding is "enriched" with some contextual information. This is done because bag-of-words generally do not embed any contextual meaning behind the words, and merely generating the most probable words for each image region wouldn't be useful. Incorporating contextual information to the image-sentence embedding is done by training a bidirectional recurrent neural network - a recurrent neural network that not only passes information forward - but also backwards. The main idea is to have the network learn an embedding that respects spatial alignment, i.e a set of words that best forms a description is one such that adjacent words best describing certain image features maps to contiguous visual blocks. These alignments serve as a context and is used as part of the traditional decoder architecture to generate novel descriptions. Karpathy and Fei-Fei 2015

III. Technical Approach and Model

A. Logistic Regression and Support Vector Machines

We looked at some attempts to generate image captions prior to the introduction of deep representation learning. We looked at Farhadi et al. 2010 in particular which attempts to relate images and sentences by creating an intermediate representation space called a "meaning" space and mapping features from both images as well as sentences to this space. These mappings are then used to define similarities between images and sentences. The images are run through various object detection algorithms and features are generated accordingly. The sentences are transformed into the meaning space by extracting a set of triplets from each sentence, comprising of an *object*, an *action* and a *scene*. The problem then reduces to predicting a triplet from a set of image features, which can represented as a (small) multi-label Markov random field.

The extraction of features from both images and sentences is achieved by using a number of potential functions. However, due to lack of time in our project, we were unable to experiment with these features. So, instead we resort to using the features generated by the Convolutional Neural Network (as explained in the subsequent section) that we modeled for our deep learning part and we only focused on the task of mapping sentences to their respective triplets. The computation of the triplets from a sentence involves analyzing the input sentence using a dependence parser and retrieve the appropriate dependency and POS (parts of speech) tags using a set of rules. In our experiment, we used the Stanford NLTK Dependency Parser to get the dependency tree. We then used a method similar to the paper, by extracting relations like the subject `nsubj`, direct object `dobj`, and other `nmod` dependencies. The scene was generally extracted from the head nouns following the prepositions `prep`. For example, a dependency parse of a sentence like

A boy wearing a red t-shirt is running through woodland . would relate to getting relations like `nsubj(running-8, boy-2)`, `nmod(running-8, woodland-10)`, etc. The resulting triplets inferred from these relations would be - Object: *boy*, Action: *running* , Scene: *woodland*.

Due to time limitations, we decided to simplify the learning task a bit. Instead of learning a triplet associated with a set of features and then generating captions based the generated triplets, we decided to treat each element of the triplet as an independent output and trained the classifier to predict that. This resulted in three independent classifiers, one for *object*, one for *action* and one for *scene*. Each of these classifiers were experiment with two types of Models - Multi-Class Logistic Regression and Support Vector Classifiers (SVC). For the latter, we attempted to use three different types of pre-defined kernels, namely the RBF (Radial Basis Function) Kernel (typically known as the *Gaussian* Kernel), Polynomial Kernel and the Linear Kernel.

B. LSTM and Gated Recurrent Units

The next approach we looked at was to use the Recurrent Neural Network to predict image captions. The model that we tried to implement is closely related to Vinyals et al. 2015. The paper describes the following architecture of a Long Short-Term Memory Network (LSTM), a specific type of Recurrent Neural Network aimed at remembering long-term dependencies. Figure 1 shows the LSTM architecture based on the paper. The equations of the gates - input i, forget f, cell c and output o gates and of the LSTM states - the memory state m and the next word predicted p are as follows:

$$i_t = \sigma(W_{ix}x_t + W_{im}m_{t-1}) \quad (1)$$
$$f_t = \sigma((W_{fx}x_t + W_{fm}m_{t-1}) \quad (2)$$
$$o_t = \sigma(W_{ox}x_t + W_{om}m_{t-1}) \quad (3)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot h(W_{cx}x_t + W_{cm}m_{t-1}) \quad (4)$$
$$m_t = ot \odot ct \quad (5)$$
$$p_{t+1} = Softmax(m_t) \quad (6)$$

The memory state m_t is used to feed to a Softmax to produce a probability distribution p_t over all words. Given I as the input image, we use a Convolutional Neural Network (CNN) to extract the feature vectors. Then we take the corresponding caption sentence which is represented as a sequence of words $S = (S_0, \ldots, S_N)$ where S_0 is a special start-of-sentence tag and S_N is an end-of-sentence tag. We then represent each word as a one-hot vector S_t of dimension equal to $|V|$, where V is the vocabulary. We then represent both sentence and image features in a single high-dimensional Manifold M and obtain an embedding of dimension r, with the corresponding points belonging to $\mathbb{R}^r$. This embedding is fed as an input x_t to the LSTM which then predicts the next word in the sequence.

$$p_{t+1} = LSTM(x_t), t \in 0, \ldots, N-1$$

where N is the maximum length of the sequence. The full architecture of CNN plus LSTM is shown in Figure 2. We also repeated this experiment with another type of RNN called a Gated Recurrent Unit, which, among other changes, combines the input and forget gate into a single "update" gate and also combines the cell state c and the memory state m.

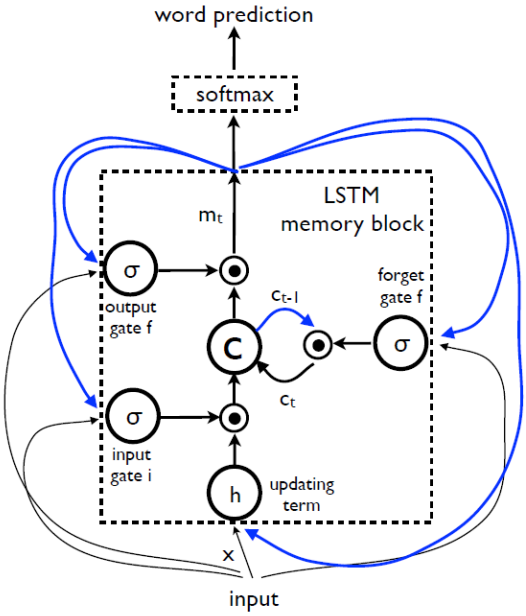


Fig. 1: General Architecture of an LSTM based on Vinyals et al. 2015

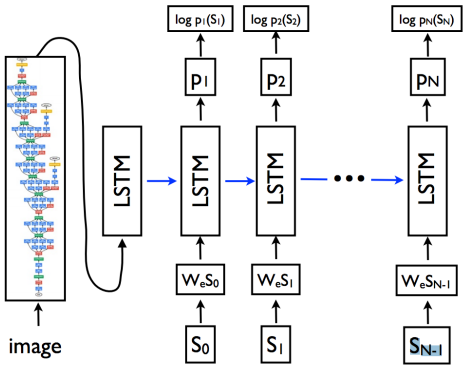


Fig. 2: Image Captioning using CNN and LSTM. The layer before LSTM is the the CNN layer where image features are obtained. The LSTM takes the features and "translates" using the learned multimodal embedding and "unrolls" the sentence as output. Vinyals et al. 2015

C. Attention based LSTM Network

The main goal of this project was to experiment with an LSTM network coupled with a feature known as *Attention Mechanism*. Based on Xu et al. 2015, attention mechanism allows the LSTM to choose what part of the image to focus

on while generating the next word in the sequence. In other words, the network can "attend" to only specific features of the image while predicting different words, hence the name "Attention Mechanism". As in the case of normal LSTM, the output caption y is a sequence of encoded words, each of which is represented as a vector of dimension $|V|$.

$$y = \{y_1, \ldots, y_C\}, y_i \in R^{|V|}$$

where V is the vocabulary and C is the maximum length of the caption (padded with zeroes where necessary). The experiment is also quite similar to the previous one and consisting of two stages as described below:

1) Encoder: We use a Convolution Neural Network (CNN) that gives us the features of the image, which the paper calls as annotation vectors. We extracted annotation features from the lower convolution layer of the network, as described in the paper. Having done this, we get L annotation vectors, each of which is a D-dimensional representation corresponding to a part of the image.

$$a = \{\mathbf{a}_1, \ldots, \mathbf{a}_L\}, \mathbf{a}_i \in \mathbb{R}^D$$

2) Decoder: The Decoder is the LSTM network itself, coupled with Attention Mechanism. The model is based on the one in the paper, which in turn is based on Zaremba et al. 2014. The architecture for this LSTM is shown in Figure 3.

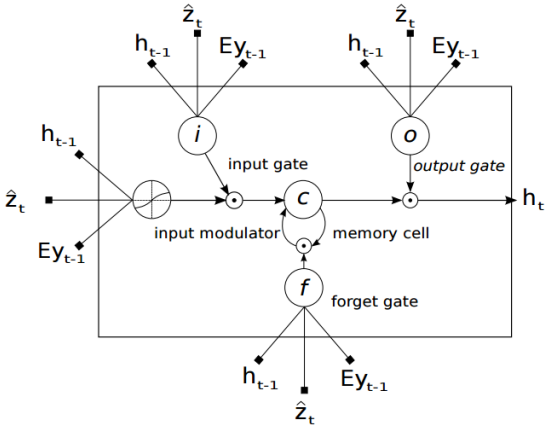


Fig. 3: Attention based LSTM Xu et al. 2015

This architecture can be expressed mathematically as follows:

$$\begin{pmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{o}_t \\ \mathbf{g}_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{pmatrix} T_{D+m+n,n} \begin{pmatrix} \mathbf{E}\mathbf{y}_{t-1} \\ \mathbf{h}_{t-1} \\ \hat{\mathbf{z}}_t \end{pmatrix} \tag{7}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \tag{8}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot tanh(\mathbf{c}_t) \tag{9}$$

In the above equations, the terms, $\mathbf{i}_t, \mathbf{f}_t, \mathbf{c}_t, \mathbf{o}_t, \mathbf{h}_t$ are the input, forget, memory, output and hidden state of the LSTM respectively. $\mathbf{E} \in R^{m*K}$ is an embedding matrix and m and n denotes the embedding and LSTM dimensionality respectively and σ and $\odot$ are the logistic sigmoid activation and element-wise multiplication respectively. $T_{n,m} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is an affine transform ($Wx + b$ for some W and b). The vector $\hat{\mathbf{z}}_t \in R^D$ is referred to as the *context vector*. This context vector helps us to focus on different parts of the image by dynamically changing the representation of different parts of the image. This can be achieved as follows: for each of the annotation vectors $\mathbf{a}_i, i = 1, \ldots, L$. we generate a positive weight α_i which can be interpreted either as the probability that location i is the right place to focus for producing the next word (known as the *hard* attention mechanism), or as the relative importance to give to location i in blending the a_i's together (known as the *soft* attention mechanism). For this experiment, we focus on soft attention. Our Model essentially follows the same convention as that in the paper, which in turn followed the convention used in Bahdanau et al. 2014. The weight α_i of each annotation vector a_i is computed by an attention model f_{att} for which we use a dense neural network conditioned on the previous hidden state h_{t1}.

$$e_{ti} = fatt(a_i, h_{t-1}) \tag{10}$$

$$\alpha_{ti} = \frac{exp(e_{ti})}{\sum_{k=1}^{L} exp(e_{tk})} \tag{11}$$

The context vector is calculated from the annotation vectors a_i and weights α_i as follows:

$$\hat{\mathbf{z}}_t = \phi(\{a_i\}, \{\alpha_i\}) \tag{12}$$

For the soft attention model we can take the expectation of the context vector $\hat{\mathbf{z}}_\mathbf{t}$ as mentioned in the paper:

$$\mathbb{E}_{p(s_t|a)}[\hat{\mathbf{z}}_t] = \sum_{i=1}^{L} \alpha_{t,i} a_i \tag{13}$$

The attention mechanism is then defined by the following

$$\hat{\mathbf{z}}_t = \phi(\{a_i\}, \{\alpha_i\}) = \sum_{i=1}^{L} \alpha_i a_i \tag{14}$$

The initial state $\mathbf{c}_0$ and the initial hidden states $\mathbf{h}_0$ of the LSTM are predicted by an MLP which is given as input the mean of the annotation vectors.

$$\mathbf{c}_0 = f_{init,c}\left(\frac{1}{L}\sum_{i}^{L} a_i\right)$$

$$\mathbf{h}_0 = f_{init,h}\left(\frac{1}{L}\sum_{i}^{L} a_i\right)$$

We then compute the output word probability with the softmax of the last layer. The output is predicted provided we are given the LSTM state, the context vector and the previous word.

$$p(\mathbf{y}_t|\mathbf{a}, \mathbf{y}_1^{t-1}) \propto \exp(\mathbf{L}_o(\mathbf{E}\mathbf{y}_{t-1} + \mathbf{L}_h\mathbf{h}_t + \mathbf{L}_z\hat{\mathbf{z}}_t)) \tag{15}$$

Where $L_o \in \mathbb{R}^{K*m}, L_h \in \mathbb{R}^{m*n}, L_z \in \mathbb{R}^{m*D}$, and E are learned parameters initialized randomly. We initialize the weights from a normal distribution.

IV. Experiments

A. Dataset

We tested on one of the popular image-captioning datasets - Flickr8k. Flickr8K dataset consists of 8000 images, each with 5 corresponding annotated captions. As an alternative, we also attempted to train to our Attention Model on Flickr30k.The majority of captions in Flickr30K consist of very verbose descriptions with an average length of 38 words. Because of the highly verbose nature of the annotated captions, we did not face much success in getting grammatically correct sentences. Also, given the time limitations, it wasn't possible for us to obtain features for the entire training set, hence we're not reporting the results here. For the Flickr8k dataset we used 6000 images for training, 1000 images for validation and the remaining 1000 images for test.

B. Preprocessing

We did some basic data pre-processing that helped speed up our training. We prepended a `#START#` tag to each sentence to indicate the start of a sequence and also appended an `#END#` tag to indicate the end of the sequence. We then constructed a vocabulary by taking words from all the sentences along with the `#START#` and `#END#` tags, and also a `#NULL#` to represent the padded zeroes at the end the of sequence. We used the vocabulary to build two dictionaries, one a word to index mapping and the other an index to word mapping. The vocabulary size derived from Flickr8k captions used for this training was 4600 words. And the maximum length of a caption was chosen to be 35.

For pre-processing images, we used the `scikit-image-transform` library. Initially we resize the image to a 256x256 size and then we crop it to match the $224 \times 224 \times 3$ dimensions. We then mean normalize each of the RGB components so that we can get a less sparse set of values. This approach to pre-processing was found in a number of implementation with similar approaches we viewed, which prompted us to choose this.

C. Details of Approach

We describe the approach of our experiments with representation learning, first in the case of a normal LSTM (and GRU) and second in the case of Attention based LSTM. In both cases, we used 8000 images overall (taken from Flickr8k) with 6000 images for training, 1000 images for validation and 1000 images for testing. The features for each of the method were computed using 16-layer Oxfordnet VGG Convolutional Neural Network (CNN) with pre-trained weights.

1) LSTM and GRU: For this model, we create sequence-based representation of the input captions and the output of the *next word* in the sequence, both of dimensions $No\_of\_samples * n - 1 * len(vocab)$, where n represents the max sequence length or the dimensionality of the LSTM. This representation along with the image features form our input sequence. This input sequence is then fed to embedding layer which produces an output embedding of dimension $r = 256$. We then add a dropout layer with dropout value of 0.5. This is then passed on to an LSTM layer. We then apply another dropout layer after this. Then a Dense layer with a softmax activation is applied to translate it back to the size of the vocabulary and to get the probabilities. We batch processed the training samples and the loss function we used was categorical cross entropy. Our optimization algorithm was Stochastic Gradient Descent (SGD) with a learning rate of 0.001.

2) Attention based LSTM: Figure 4 shows the overview of the Attention based LSTM Model. Similar to the previous model, we use a VGG-16 standard CNN model to get the image features. For the Attention based model however, we pulled the results from a lower convolutional layer instead of the final CNN output layer. These results become our annotation vectors. The original set of annotation vectors$\{a_i\}$ was of shape $14 \times 14 \times 512$, which we flattened out to a shape of 196×512, i.e. we had $L = 196$ annotation vectors each of dimension $D = 512$. Then, similar to the previous case, we create a sequence based representation of our input captions and the output of the *next word* in the sequence of dimensions $No\_of\_samples * n - 1 * len(vocab)$, n being the max sequence length or the dimensionality of the LSTM. The captions were passed to a Time-distributed dense layer of dimension $D = 512$ and sequence length $n - 1$. This layer was then merged with the annotation vectors to form an input sequence, which was then passed to the Attention based LSTM. Unlike the normal LSTM, this LSTM also has an extract context state. Both the hidden state h and the context state z were initialized (as explained in Section III-C) through two independent Multilayer Perceptrons fed with an initial value equal to the mean of the annotation vectors and weights initialized randomly from a normal distribution. We then compute a soft max layer to get the probabilities and then output the word with the highest probability at a given time sequence. The loss function we used was `categorical cross-entropy` and the optimization algorithm used was Root Mean-Square Propagation (rmsprop), same as in the paper.

D. Evaluation Metrics

The evaluation metrics we used were from standard machine translation metrics, and they were as follows:

BLEU: This involves finding out the number of n-gram overlaps. We used the modified precision method of BLEU available in NLTK 3.0, where it tries to remove some of the deficiencies of the normal BLEU metric by not taking into account an n-gram once it is used/exhausted.

METEOR: This was a metric introduced in Lavie 2014 which helps take into account the things for which the BLEU metric tends to give false positive results. The METEOR metric also takes into account the stem words for similar matching. In addition to that, it also takes into account similar words and paraphrased words and does not penalize them.

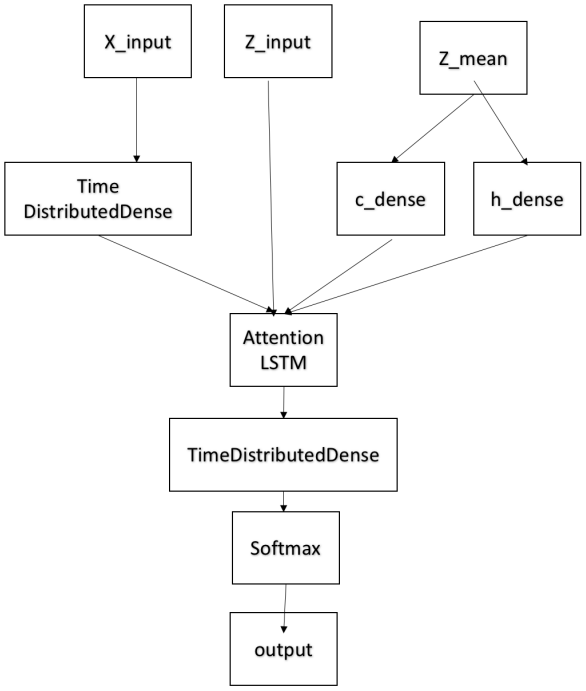


Fig. 4: Overview of the Attention based LSTM Model

CIDER: This is a relatively new metric introduced in Vedantam et al. 2015, wherein it tries to improve upon the procedures used by METEOR by factoring in the TF-IDF scores for the correct matches. This helps it do better for the cases where the reference as well as predicted text is short, only 1-2 sentences long, which is very much the case with image captioning. There is alternative version of CIDEr called CIDEr-D, which tries to solving the problem of "Gaming" associated with the above metrics, wherein a sentence rated poorly by humans can get high score. This is achieved, among other ways, by the removal of stemming and introducing a Gaussian penalty based on difference in the lengths between reference and the generated sentences.

E. Quantitative analysis of model results

Here, we present the results of our models as evaluated by the different metrics (for the deep learning case). For the case of non-deep learning, since we did not have the time to generate sentences from the triplets, we could not use the evaluation metrics to evaluate our results, so we instead report the training and test accuracies.

1) Non Deep Learning Methods: The training set for this method was 900 samples from Flickr8k and the test set was 90 samples from flickr8k with the triplets of the Object, Action and Scene taken from our algorithm to get the triplets given a caption. We tried Multi-class Logistic Regression and Support Vector Classifier (SVC) Machines with Linear, Polynomial (degree 3) and RBF (Gaussian) Kernel. We report the following training and test accuracies in Table I. Despite accounting for the limitations in our method (as mentioned in Section III-A), we found that this methodology did not work as well as we expected, with Logistic Regression and Linear Kernel based SVMs giving semi-decent results and SVMs based on other Kernels performing worse.

TABLE I: Accuracies for LR and SVM

Method	Training accuracy	Test accuracy
LR - Object	**31**	**27**
LR - Action	**13**	**8**
LR - Scene	**10**	**2**
SVM(Linear kernel) - Object	**16**	**29**
SVM(Linear kernel) - Action	**11**	**8**
SVM(Linear kernel) - Scene	**2**	**2**
SVM(RBF kernel) - Object	16	14
SVM(RBF kernel) - Action	11	7
SVM(RBF kernel) - Scene	2	0
SVM(Polynomial kernel) - Object	15.9	13
SVM(Polynomial kernel) - Action	10.9	7
SVM(Polynomial kernel) - Scene	1.1	0

We represent all the metrics above in the percentage Scale. LR represents Logisitc Regression. SVM represents Support Vector Machines.

2) Deep Learning Methods: Table II shows the performance of the LSTM, GRU and Attention based LSTM models as evaluated by the three metric described in Section IV-D. Even though, we weren't able to get results comparable to Xu et al. 2015, the evaluation still shows that Attention based model performs better in most cases than normal LSTM or GRU. Figure 5 shows the variance of training loss with the number of iterations for the case of the normal LSTM Model. Figure 6 shows a similar plot for the case of Attention based LSTM (the line in the front shows the moving average). A plot of training accuracy is shown in shown in Figure 7.

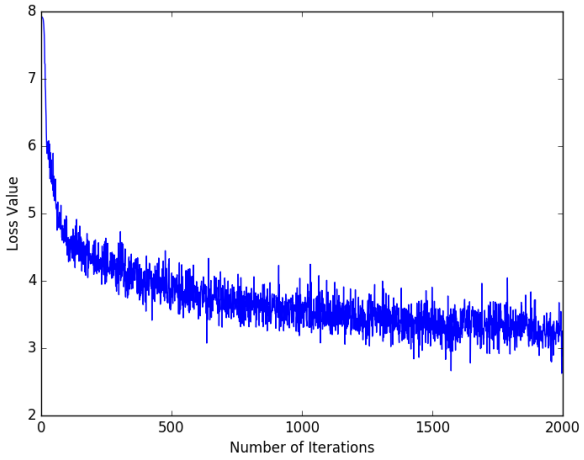


Fig. 5: Loss graph during training for LSTM model

			BLEU					CIDER	
Dataset	**Features**	**Model**	**BLUE-1**	**BLUE-2**	**BLUE-3**	**BLUE-4**	**METEOR**	**CIDER**	**CIDER-D**
Flickr8k	VGG-16	LSTM	**59**	22	7	1.4	6.8	1.4	1
		GRU	55	**23**	8	3	10	3	2.2
		Attention	49	22	**10**	**4.7**	**17**	**13**	**11**

TABLE II: Evaluation Results For LSTM, GRU and Attention Models

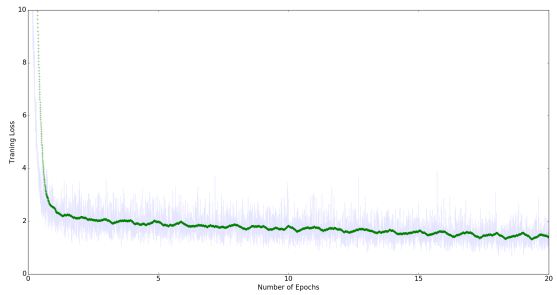


Fig. 6: Graph of training loss for Attention based LSTM

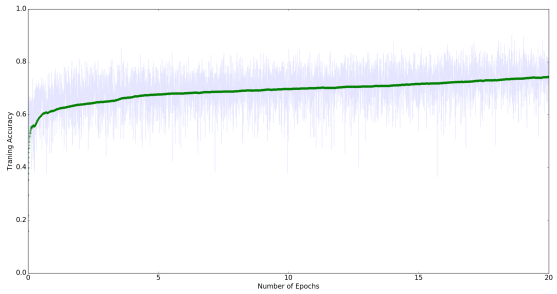


Fig. 7: Graph of training accuracy for Attention based LSTM

F. Qualitative analysis - Image Captions and Visualizing weights for the Attention based Model

Some of the sample captions (along with their images) generated by the LSTM/GRU Models and the Attention-based LSTM Model are shown in Appendix I. Along with this, we also tried to visualize the weights for the annotation vectors learned by the Attention based LSTM similar to Xu et al. 2015. The inputs to the CNN are 224x224 images and the outputs are 14x14 annotation vectors each of dimension 512. So, to visualize the attention weights for the soft model, we simply upsample the weights by a factor of $2^4 = 16$ and apply a Gaussian filter, as done in the paper. Appendex II shows the results for some images. It can be inferred that this visualiation helps in qualitatively evaluating the results of the Attention based LSTM in comparison with other methods, in that it can be seen from the images that the model is (generally) looking at places where a human would tend to look while generating (speaking) the corresponding word in the caption. Also, for the case that the model failed to predict the correct word, visualization gives us an insight into where the model went wrong in the process. As mentioned in the paper, this approach is better than the previous work done in Karpathy and Fei-Fei 2015 for instance, which relies on detecting "objects" in images such as faces, buildings, etc. whereas the Attention mechanism allows the model to learn those features that are not necessarily objects yet play a key role in describing the image.

V. CONCLUSIONS

As a part of this Project [1], we tried to explore various approaches that are and have been used for generating captions for Images. For generating image features, we used a Convolutional Neural Network and extracted annotation vectors to be fed to the later models. We started with simpler methods like Logistic Regression and Support Vector Machine Classifiers and saw how they aren't suited for the task since they require a lot of human intervention and hand-picked design of features for them to produce decent results. We then turned to representation learning, starting with a kind of Recurrent Neural Network called a Long Short-Term Memory Network which excels at remembering long-term dependencies, and we did get close to the results mentioned in Xu et al. 2015. We also tried another variant of RNN called a Gated Recurrent Unit, which also gave a performance similar to LSTM. We then moved to our primary objective of the project, which is to experiment with an LSTM coupled with Attention Mechanism. Out of the two techniques mentioned in the paper, we chose soft attention mechanism and experimented with it. We report our results in three different metrics, namely BLUE, METEOR and CIDER. The results show that the Attention based LSTM does better in most cases than the normal LSTM or GRU network. We also tried to visualize some of the results generated by the Attention based LSTM network in order to give an insight into what the network sees and what it decided to focus on. The visualizations showed that alignments learned by the network have a good correspondence with human intuition in most cases. We believe that attention mechanism is the key to generating better image captions and we hope that future improvements in capturing attention will likely generate better captions for images.

[1]The source code for this Project is available at https://github.com/danielsamfdo/image_caption_using_attention

REFERENCES

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. Every picture tells a story: Generating sentences from images. In *European Conference on Computer Vision*, pages 15–29. Springer, 2010.

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

Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137, 2015.

Ryan Kiros, Salakhutdinov Ruslan, and Richard S. Zemel. Unifying visual-semantic embeddings with multimodal neural language models. 2014.

Michael Denkowski Alon Lavie. Meteor universal: language specific translation evaluation for any target language. *ACL 2014*, page 376, 2014.

Bryan A Plummer, Liwei Wang, Chris M Cervantes, Juan C Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2641–2649, 2015.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4566–4575, 2015.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164, 2015.

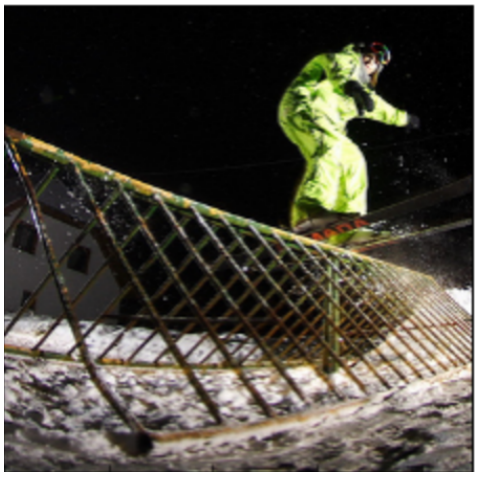
Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2(3):5, 2015.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

APPENDIX I
SAMPLES OF CAPTIONS GENERATED BY OUR MODEL

A. *LSTM/GRU Model*

A man in a blue jacket is standing on a red slide



A man in a white shirt is running through a snow

B. *Attention Model*

A men in hard hats are operating a denim tree ground

A men are a men are playing the guitar and the man man playing in him

A young child wearing a blue shirt is on a red area in a red ball on

A girl with her hair holding her bottle with red cap

APPENDIX II
VISUALIZATION OF WEIGHTS FOR ATTENTION BASED LSTM

In this appendix, we present visualization of weights for some images and captions predicted by the Attention based LSTM Model. The *white* region on each images indicates the regions where the model roughly attends to.

Caption - *A dog leaps into the air in a grassy field surrounded by trees*

Caption - *Two men walking on a bridge sky in the background*

Caption - *A men in hard hats are operating a denim tree ground*

Caption - *A young child wearing a blue shirt is on a red area in a red ball on*

Caption - *glass are standing on helmets in the background with a building sky*