Image Captioning using Neural Networks

Arpit Arora, Lijo Thomas, Prashant Singh, Tanveer Rainu

Final Project-IDS 576
Masters in Science of Business Analytics
University of Illinois at Chicago

Abstract

Automatically generating textual description from an artificial system given any image is the task of image captioning. Image captioning requires to recognize the important objects, their attributes and their relationships in an image. It also needs to generate syntactically and semantically correct sentences. Deep learning-based techniques are capable of handling the complexities and challenges of image captioning. In this project, we aim to present a comprehensive review of one of the best-performing existing deep learning-based image captioning technique using an encoder decoder architecture using CNNs and LSTMs. We discuss the foundation of this technique to analyze its performances, strengths and limitations. We also discuss in detail the preprocessing, modeling and the evaluation pipeline used in this deep learning based automatic image captioning solution.

1.Introduction

Automatically describing the content of an image is a fundamental problem in artificial intelligence that connects computer vision and natural language processing. Image captioning is important for many reasons. For example, they can be used for automatic image indexing. Image indexing is important for Content-Based Image Retrieval (CBIR) and therefore, it can be applied to many areas, including biomedicine, commerce, the military, education, digital libraries, and web searching. Social media platforms such as Facebook and Twitter can directly generate descriptions from images. The descriptions can include where we are (e.g., beach, cafe), what we wear and importantly what we are doing there. Image captioning is a popular research area of Artificial Intelligence (AI) that deals with image understanding and a language description for that image. Image understanding needs to detect and recognize objects. It also needs to understand scene type or location, object properties and their interactions. Generating well-formed sentences requires both syntactic and semantic understanding of the language The solution offered in this project uses an encoder-decoder architecture- so basically, the task of image captioning is divided into two modules logically – one is an image based model – which extracts the features and nuances out of our image creating a feature vector capturing high level features in the image, and the other is a language based model (using RNN in conjunction with LSTM)– which translates the features and objects given by our image based model to a natural sentence.

Understanding an image largely depends on obtaining image features. The techniques used for this purpose can be broadly divided into two categories: (1) Traditional machine learning based techniques and (2) Deep machine learning based techniques. In traditional machine learning, hand crafted features such as Local Binary Patterns (LBP), Scale-Invariant Feature Transform (SIFT), the Histogram of Oriented Gradients (HOG), and a combination of such features are widely used. In these techniques, features are extracted from input data. They are then passed to a classifier such as Support Vector Machines (SVM) in order to classify an object. Since hand crafted features are task specific, extracting features from a large and diverse set of data is not feasible. Moreover, real world data such as images and video are complex and have different semantic interpretations.

On the other hand, in deep machine learning based techniques, features are learned automatically from training data and they can handle a large and diverse set of images and videos. For example, Convolutional Neural Networks (CNN) are widely used for feature learning, and a classifier such as Softmax is used for classification. CNN is generally followed by Recurrent Neural Networks (RNN) in order to generate captions.

Deep neural network techniques using the convolutional neural network algorithms have successfully solved image classification problems with astounding accuracy. Convolutional neural networks (CNN) is a special architecture of artificial neural networks, proposed by Yann LeCun in 1988. On 30 September 2012, a convolutional neural network (CNN) called AlexNet achieved a top-5 error of 15.3% in the ImageNet 2012 Challenge, more than 10.8 percentage points lower than that of the runner up. This was made feasible due to the utilization of Graphics processing units (GPUs) during training, an essential ingredient of the deep learning revolution. In 2017, 29 of 38 competing teams in the imageNet challenge had greater than 95% accuracy.[1] All of these pretrained networks are freely available on github and can be used for image related tasks incorporating CNNs using transfer learning. Transfer learning allows us to train deep networks using significantly less data then we would need if we had to train from scratch. With transfer learning, we are in effect transferring the "knowledge" that a model has learned from a previous task, to our current one. The idea is that the two tasks are not totally disjoint, and as such we can leverage whatever network parameters that model has learned through its extensive training, without having to do that training ourselves. Although we are not using CNNs for image recognition here, we can use the pretrained CNN architecture as a feature extractor which captures all the high-level features of the image. The penultimate layer giving us a feature vector capturing high level features of the image and we get rid of the last SoftMax layer to classify the images and instead feed the resultant feature vector to a language model. The decoder part takes care of converting the encoded vector capturing high level features into text using RNNs in with LSTMs. Text Generation is a type of Language Modelling problem. Language Modelling is the core problem for a number of natural language processing tasks such as speech to text, conversational system, and text summarization. A trained language model learns the likelihood of occurrence of a word based on the previous sequence of words used in the text. Unlike Feed-forward neural networks in which activation outputs are propagated only in one direction, the activation outputs from neurons propagate in both directions (from inputs to outputs and from outputs to inputs) in Recurrent Neural Networks. This creates loops in the neural network architecture which acts as a 'memory state' of the neurons. This state allows the neurons an ability to remember what have been learned so far. The memory state in RNNs gives an advantage over traditional neural networks but a problem called Vanishing Gradient is associated with them. In this problem, while learning with a large number of layers, it becomes really hard for the network to learn and tune the parameters of the earlier layers. To address this problem, A new type of RNNs called LSTMs (Long Short Term Memory) Models have been developed. LSTMs have an additional state called 'cell state' through which the network makes adjustments in the information flow. The advantage of this state is that the model can remember or forget the leanings more selectively. The details of LSTM are beyond the scope of this report, but the LSTM architecture brings with itself a special advantage that it can also incorporate "memory" in the algorithm. The architecture used for the image captioning problem looks something like this:

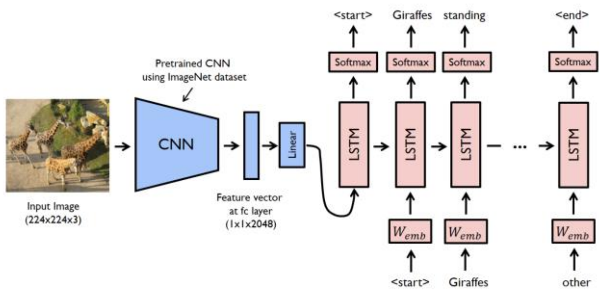
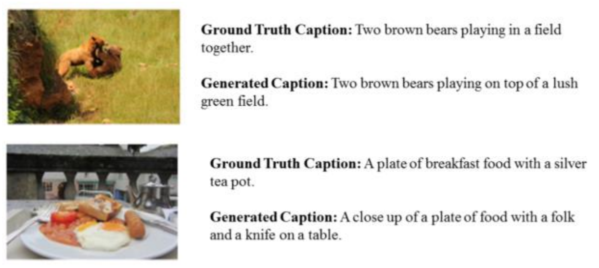


Fig1: Architecture used for image captioning

2.Dataset

MS COCO Dataset: Microsoft COCO Dataset [2] is a very large dataset for image recognition, segmentation, and captioning. There are various features of MS COCO dataset such as object segmentation, recognition in context, multiple objects per class, more than 300,000 images, more than 2 million instances, 80 object categories, and 5 captions per image. Many image captioning methods [4,5] use the dataset in their experiments. For example, Wu et al. [6] use MS COCO dataset in their method and the generated captions of two sample images are shown in Figure2 below:



Ground Truth Caption: Two brown bears playing in a field together.

Generated Caption: Two brown bears playing on top of a lush green field.

Ground Truth Caption: A plate of breakfast food with a silver tea pot.

Generated Caption: A close up of a plate of food with a folk and a knife on a table.

Fig 2. Captions generated by Wu et al. [6] on some sample images from the MS COCO dataset.

For our project we used a subset of the MS-COCO dataset. We have 82,783 images with captions for training and 40,504 images for validation. The data was extracted using python coco API [12]. The COCO API assists in loading, parsing, and visualizing annotations in COCO. The API supports multiple annotation formats. We followed the following github python API demo to extract the data [13].

3.Related Work

A lot of research work has been done in the image captioning space [1]. The techniques used in image captioning can be divided into three categories. The categories include template-based image captioning, retrieval-based image captioning, and novel caption generation. Template-based approaches have fixed templates with a number of blank slots to generate captions. In these approaches, different objects, attributes, actions are detected first and then the blank spaces in the templates are filled. Template-based methods can generate grammatically correct captions. However, templates are predefined and cannot generate variable-length captions. Some of the template based approaches have been cited in the references [7,8]. Captions can be retrieved from visual space and multimodal space. In retrieval-based approaches, captions are retrieved from a set of existing captions. Retrieval based methods first find the visually similar images with their captions from the training data set. These captions are called candidate captions. The captions for the query image are selected from these captions pool. These methods produce general and syntactically correct captions. However, they cannot generate image specific and semantically correct captions. Some of the work in this domain has been cited in the references [9, 10]

Novel captions can be generated from both visual space and multimodal space. A general approach of this category is to analyze the visual content of the image first and then generate image captions from the visual content using a language model. These methods can generate new captions for each image that are semantically more accurate than previous approaches. Most novel caption generation methods use deep machine learning based techniques. Therefore, deep learning based novel image caption generating method is the one we chose for our project.

Novel caption generation-based image caption methods mostly use visual space and deep machine learning based techniques. Captions can also be generated from multimodal space. Deep learning-based image captioning methods can also be categorized on learning techniques: Supervised learning, Reinforcement learning, and Unsupervised learning. In supervised learning, training data come with desired output called label. Unsupervised learning, on the other hand, deals with unlabeled data. Generative Adversarial Networks (GANs) are a type of unsupervised learning techniques. Reinforcement learning is another type of machine learning approach where the aims of an agent are to discover data and/or labels through exploration and a reward signal. A number of image captioning methods use reinforcement learning and GAN based approaches.

Usually captions are generated for a whole scene in the image. However, captions can also be generated for different regions of an image (Dense captioning). Image captioning methods can use either simple Encoder-Decoder architecture or Compositional architecture. There are methods that use attention mechanism, semantic concept, and different styles in image descriptions. Some methods can also generate description for unseen objects. We have used encoder-decoder architecture for this project.

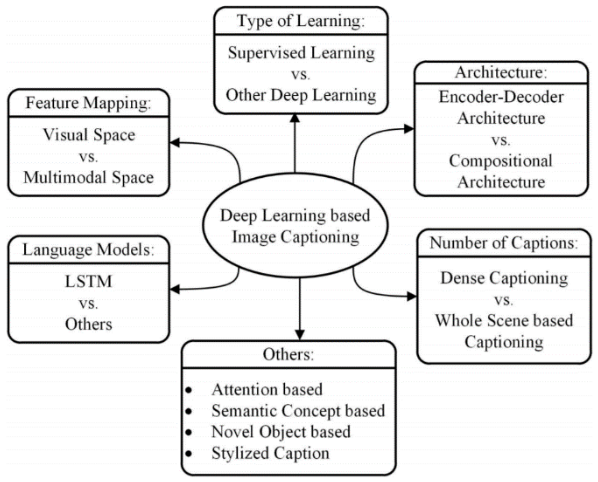

Type of Learning:
Supervised Learning vs. Other Deep Learning
Feature Mapping:
Visual Space vs. Multimodal Space
Architecture:
Encoder-Decoder Architecture vs. Compositional Architecture
Deep Learning based Image Captioning
Language Models:
LSTM vs. Others
Number of Captions:
Dense Captioning vs. Whole Scene based Captioning
Others:
Attention based
Semantic Concept based
Novel Object based
Stylized Caption

Fig3. An overall taxonomy of deep learning-based image captioning.

4.Models and Methods

In this section we will discuss the modeling steps in detail including data preprocessing which included image preprocessing, transformations and augmentation, followed by vocabulary preparation, followed by defining and training the model.

4.1 Data Preprocessing

4.1.1 Image Preprocessing

There are a number of pre-processing steps we might wish to carry out before using the dataset. We used the following preprocessing steps before going ahead and building the model.

1) Uniform aspect ratio: One of the first steps we took was to ensure that the images have the same size and aspect ratio. Most of the neural network models assume a square shape input image, which means that each image needs to be checked if it is a square or not, and cropped appropriately. Cropping is done to select a square part of the image however we used random crop to incorporate some stochasticity in our model and do some data transformation.

2) Image Scaling: Once we've ensured that all images are square (or have some predetermined aspect ratio), we scaled each image appropriately. We've decided to have images with width and height of 256 pixels. There are a wide variety of up-scaling and down-scaling techniques and we used a library function to do this for us.

3) Normalizing the images: Data normalization is an important step which ensures that each input parameter (pixel, in this case) has a similar data distribution. This makes convergence faster while training the network. Data normalization is done by subtracting the mean from each pixel and then dividing the result by the standard deviation. The distribution of such data would resemble a Gaussian curve centered at zero.

4) Random horizontal flipping: Horizontally flip the given PIL Image randomly with a given probability. In pytorch, we have to set the probability of the image being flipped and the default value for this parameter is 0.5- we used the default parameter.

4.1.2 Vocabulary Preparation

One of the first things required for image captioning task is to have a powerful vocabulary. To create the vocabulary we split all the training captions into unique words of meaning. This process of splitting the documents into meaningful words is called tokenization. Tokenization is a powerful tool, it breaks the unstructured text, into chunks of information which can be counted as discrete elements thus representing the unstructured data into a structured form. For this project our tokens are going to be limited to words, punctuation marks and numbers, but the techniques we use are easily extended to any other units of meaning contained in a sequence of characters. The technique of tokenizing is not perfect in retaining the information content of the document. Nonetheless, this technique retains enough of the information content of the text to produce useful and interesting natural language processing models. A technique for tokenization is one-hot vector encoding. In this technique we represent numerical vector for each word in a document as a one-hot word vector. A sequence of these one-hot word vectors fully captures the original document text in a numerical data structure.

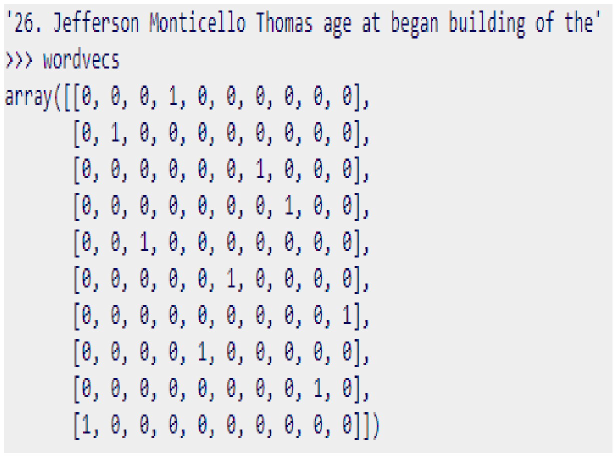

```
'26. Jefferson Monticello Thomas age at began building of the'
>>> wordvecs
array([[0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
       [0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
       [0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
       [0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
       [1, 0, 0, 0, 0, 0, 0, 0, 0, 0]])
```

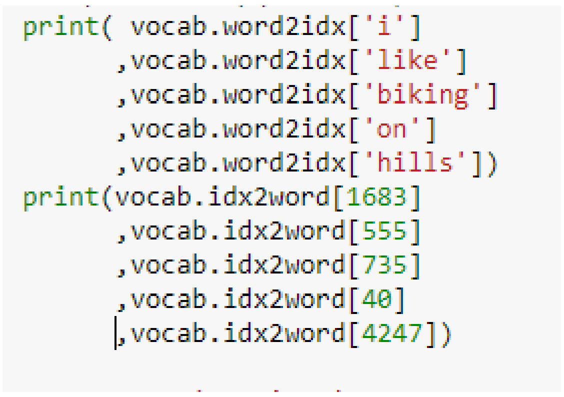
Fig 5: Example of a one-hot feature representation

In this representation of a sentence, each row or line is a word. The '1' in a column means it is the word represented by that row or vector. Because these are one hot vectors all the other positions are set to zero. In the illustration above, A one(1) means on or hot. A zero(0) means off or absent.

In this type of vector representation an advantage is no information is lost and we can get back the sentence easily from the one hot encoding. As a result one-hot word vectors are typically used in neural nets, sequence-to-sequence language models and generative language models. The drawback of this technique is it takes a lot space to store the information. In most cases there are millions of tokens, to represent these would take a lot of memory. Storing all those zeros and trying to remember the order of the words doesn't make much sense. Fortunately or unfortunately, the words in our vocabulary are sparsely utilized in any given text. So rather than creating a one hot vector representation, we can use a binary vector indicating the presence or absence of a particular word in a sentence. This vector representation of a set of sentences could be indexed to indicate which words were used in which document.

For this we use two dictionaries one with each word in the vocabulary as the key and a number as its value. The second dictionary has the number as the key and the corresponding word as the value. These two dictionaries define our vocabulary. We can encode the sentences as the values based on the first dictionary and get back the words based on the value which is the key from the second dictionary.

Lets look at this concept we used with an example:


```
print( vocab.word2idx['i']
      ,vocab.word2idx['like']
      ,vocab.word2idx['biking']
      ,vocab.word2idx['on']
      ,vocab.word2idx['hills'])
print(vocab.idx2word[1683]
      ,vocab.idx2word[555]
      ,vocab.idx2word[735]
      ,vocab.idx2word[40]
      ,vocab.idx2word[4247])
```

Output:

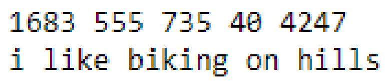

```
1683 555 735 40 4247
i like biking on hills
```

Fig 6: Creating idx2word and word2idx dictionaries

In the example above, word2idx is having each word in the vocabulary as key and a number as the value. Whereas, Idx2word has the above number as the key and corresponding word as the value.

4.2 Defining and training the model

4.2.1 Architecture implementation

The neural network-based image captioning methods work as just simple end to end manner. These methods are very similar to the encoder-decoder framework-based neural machine translation. In this network, global image features are extracted from the hidden activations of CNN and then fed them into an LSTM to generate a sequence of words. A typical method of this category has the following general steps:

(1) A vanilla CNN is used to obtain the scene type, to detect the objects and their relationships.

(2) The output of Step 1 is used by a language model to convert them into words, combined phrases that produce an image captions.

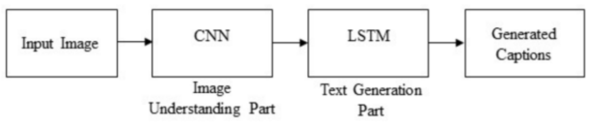


Fig 7: Block diagram of encoder-decoder based methodology used

The core idea is develop a CNN as an encoder which represents each image as a feature vector for the decoder (RNN) to detect and predict meaningful captions for the image.

For the encoder network, the natural choice is to use train a CNN. However, deeper neural networks are more difficult to train. So, we are using transfer learning to ease the training of networks that are substantially deeper and difficult to train on a CPU. For this project we are using resnet-152 which was pretrained on imageNet data. Since the resnet-152 is trained to give 1000 categories, and will have 1000*1 nodes thus In our scenario we have removed the last layer of the resnet-152 pre trained CNN, and the final layer has been converted to the embedding size using a linear transformation to correspond to the input of the decoder network. The output of the encoder CNN acts as the input for the decoder network. The network architecture can be looked at in detail at the following Kaggle page provided by Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun added to the references section [11].

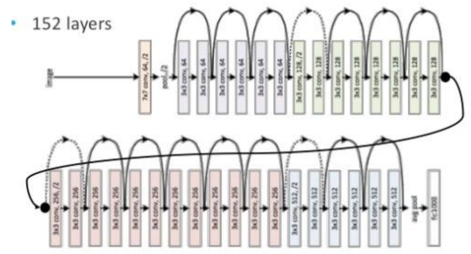


Fig 8: ResNet152 architecture snapshot

The decoder network should be trained unlike the encoder. For training the decoder network we are using a subset of the MS-COCO dataset. The decoder network is defined as follows The embedding layer: learnable parameters, which helps in the determining the similarity between the data characteristics. Parameters we used for embedding layer are size of dictionary of embeddings, the size of embedding vector. Embeddings can be trained or we can use pretrained embeddings. For our project we are training the embeddings. The size of embedding vector we choose is 256 * 1.

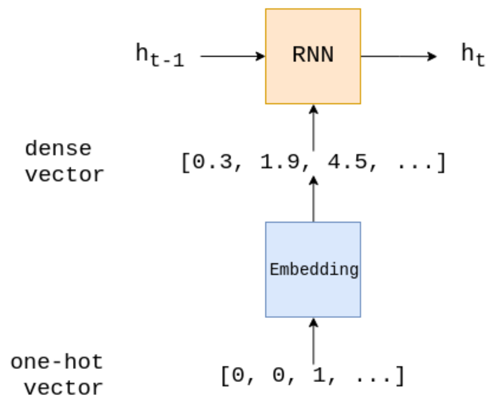


Fig 9: Feeding the caption words to the RNN network via the embedding layer

Embedding layer is followed by LSTM, which overcomes the vanishing gradient problem by having an extra recurrent state called a cell, which can be thought of as the "memory" of the LSTM - and we used multiple gates which control the flow of information into and out of the memory. The input to LSTM is embedding vector size * vocabulary size. In our case, it is (256 * 512) . Other parameters for defining the LSTM are, number of features in hidden state, number of layer in LSTM, batch_first=true (then the input and output tensors are provided as (batch, seq, feature). Further a linear transformation is applied with parameters input size & output size to get the predicted output.To sample and generate the image caption we are using greedy search method.

4.2.2 Training the model

We selected suitable loss measures, optimizer and learning rate to train our model. Discussed below is some detail around these configurations and hyperparameters.

Loss measure: We are using cross entropy for calculating the loss. Cross entropy is a good cost function for neural networks for classification tasks and uses activation functions in the output layer that model probabilities (Sigmoid) or distributions

(SoftMax). First of all it is very intuitive to work with entropy (information content) when handling probabilities. But more importantly the cross entropy has some nice properties, especially for sigmoid activation functions. When a neuron has a value close to 0 or 1, the gradient of the loss function will become very small. In turn the gradient of the whole cost function will become very small. A neuron that reached this state is called saturated and will change its weights very slowly. This can be detrimental to the learning process of a network. However, when we use cross entropy as an error term the gradient only depends on the neuron's output, the target and the neuron's input. This avoids learning slow-down and helps with the vanishing gradient problem from which deep neural networks suffer.

Optimizer: Adaptive moment estimation (ADAM) is the method we used as the optimization algorithm. Adam computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients, Adam also keeps an exponentially decaying average of past gradients.

Learning Rate: The amount that the weights are updated during training is referred to as the step size or the "learning rate". Specifically, the learning rate is a configurable hyperparameter used in the training of neural networks that has a small positive value, often in the range between 0.0 and 1.0.We have used a learning rate of 0.01 for updating the gradients.

5. Testing the Model: Generating Captions

Finally, after training our model, we now are trying to test the model results. Before testing the images the images are preprocessed just in line with our training dataset images. Shown below are the captions generated by our model.

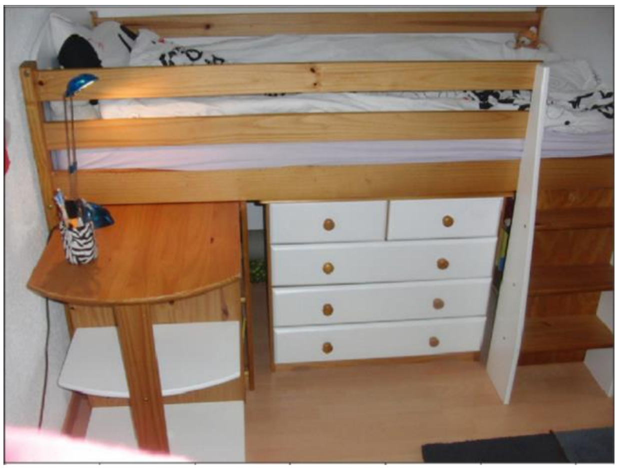


Fig 10: Test Image 1. Caption generated:
"A Wooden Table Topped With A Wooden Bunk Bed"

We see that the model performs really well and is able to capture elements in the image such as: bunk bed, wooden, table and the caption has a perfect semantic meaning and structure. Given below are some more examples of test image- caption combinations:

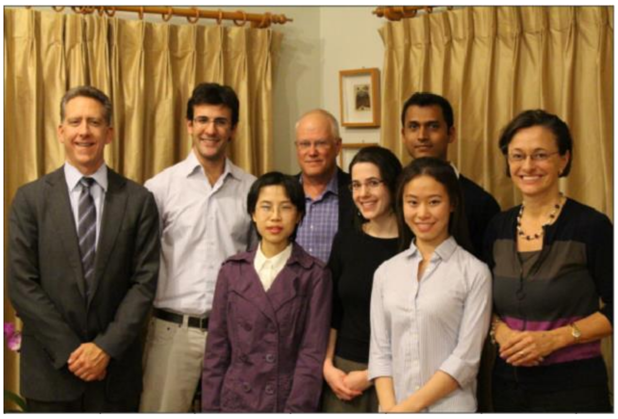


Fig 11: Test Image 2. Caption generated:
"A Group Of People Standing Around Each Other In A Room"

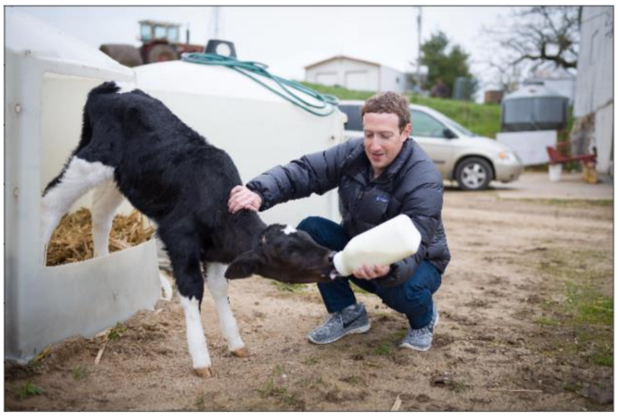


Fig 12: Test Image 3. Caption generated:
"A Man Is Standing Next To A Dog On A Leash ."

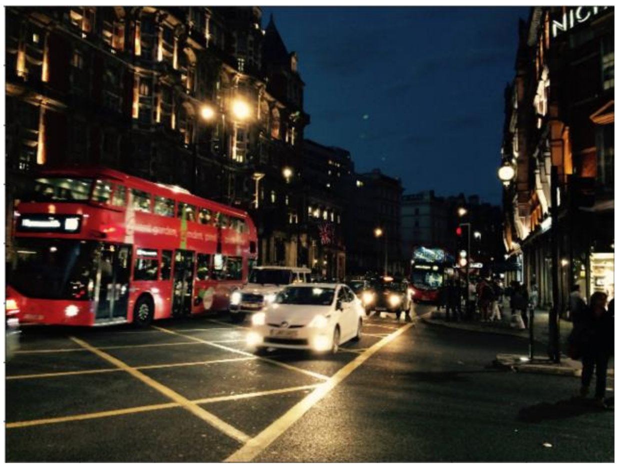


Fig 13: Test Image 4. Caption generated:
"A Red Double Decker Bus Driving Down A Street"

We see that the model performs fairly well! However, there is still some scope of improvement, lets look at some examples where the model although is able to capture the gist of the image and the elements but misses out on capturing the underlining details in the image:

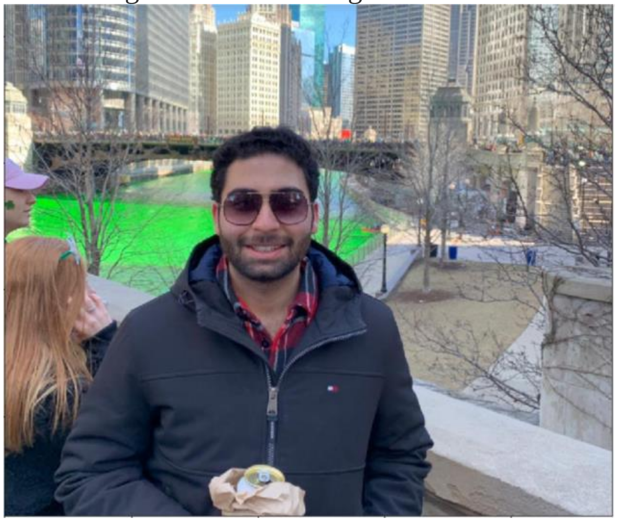


Fig 14: Test Image 5. Caption generated:
"A Woman Is Holding A Cell Phone In Her Hand"

The model fails to figure out that the person is a male. Also, although the model correctly recognizes that the person is holding something in her hand- its definitely not a cell phone.

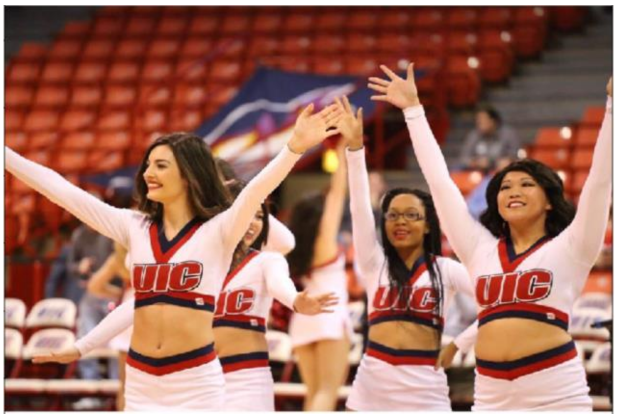


Fig 15: Test Image 6. Caption generated:
"A Group Of Men On A Field Playing With A Frisbee"

Clearly this caption is totally off from reality. We recognize that the model fails to differentiate among genders and in some cases fails to identify the scene altogether. This can be attributed to the fact that we only ran 2 epochs because of the GPU capacity and size of the dataset. The accuracy may further increase on running 5-10 epochs.

6. Challenges, Learning and Future Work

In this paper we have implemented state of the art model for performing image captioning on COCO dataset. We relied on language-based model and image-based model to implement the project. In the due course of implementation, we realized one of the major challenges was that these models do not generalize well to some of the images- the captions though are not totally off target but definitely fail to capture things such as gender, back scenes and background, etc. This happens because the model was trained to capture any tiny amount of visual concepts as compared to what a human may encounter in everyday life. Therefor it is worth exploring the use of attention encoder-decoder framework which provides a fallback option to the decoder. Also as discussed, we were only able to run 2 full epochs across the training dataset due to constraints arising from data size and gpu capability. Further, LSTM could be extended to include visual sentinel to increase its performance. Lastly, we plan to implement the latest state of the art model in the field of image captioning developed by Microsoft, Caption-Bot and compare the test performance.

7. References

[1] *MD. ZAKIR HOSSAIN, FERDOUS SOHEL, MOHD FAIRUZ SHIRATUDDIN and HAMID LAGA: A Comprehensive Survey of Deep Learning for Image Captioning*

[2]*Gershgorn, Dave (10 September 2017).* "The Quartz guide to artificial intelligence: What is it, why is it important, and should we be afraid?". *Quartz*. Retrieved 3 February 2018.

[3] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr DollÃąr, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In European conference on computer vision. Springer, 740–755.

[4] Bo Dai, Dahua Lin, Raquel Urtasun, and Sanja Fidler. 2017. Towards Diverse and Natural Image Descriptions via a Conditional GAN. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR). 2989–2998.

[5] Chuang Gan, Zhe Gan, Xiaodong He, Jianfeng Gao, and Li Deng. 2017. Stylenet: Generating attractive visual captions with styles. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 3137–3146.

[6] Qi Wu, Chunhua Shen, Anton van den Hengel, Lingqiao Liu, and Anthony Dick. 2015. Image captioning with an intermediate attributes layer. arXiv preprint arXiv:1506.01144 (2015).

[7] Ali Farhadi, Moh1sen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. 2010. Every picture tells a story: Generating sentences from

images. In European conference on computer vision. Springer, 15–29.

[8] Siming Li, Girish Kulkarni, Tamara L Berg, Alexander C Berg, and Yejin Choi. 2011. Composing simple image descriptions using web-scale n-grams. In Proceedings of the Fifteenth Conference on Computational Natural Language Learning. Association for Computational Linguistics, 220–228.

[9] Yunchao Gong, Liwei Wang, Micah Hodosh, Julia Hockenmaier, and Svetlana Lazebnik. 2014. Improving imagesentence embeddings using large weakly annotated photo collections. In European Conference on Computer Vision. Springer, 529–545

[10] Micah Hodosh, Peter Young, and Julia Hockenmaier. 2013. Framing image description as a ranking task: Data, models and evaluation metrics. Journal of Artificial Intelligence Research 47 (2013), 853–899.

[11] https://www.kaggle.com/pytorch/resnet152

[12]https://github.com/cocodataset/cocoapi

[13]https://github.com/cocodataset/cocoapi/blob/master/PythonAPI/pycocoDemo.ipynb

[14] https://github.com/yunjey/pytorch-tutorial/tree/master/tutorials/03-advanced/image_captioning