Application of Data Science
Show, Attend and Tell
Neural Image Caption Generation with Visual Attention

Sukhdeep Singh : 44442467
Mahima Gupta : 45565538
Shreeya Baidya : 45499780
Kartik Kamboj : 44448163

I. Brief description of the source paper, and justification

The paper Show, Attend and Tell: A Neural Image Caption Generation with Visual Attention published by Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel Yoshua Bengio presents a Neural Image Caption (NIC) model with visual attention, i.e., an attention-based model, that automatically learns to describe the content of an image [1]. In order to train the model, the researchers have used two alternative mechanisms, i.e. stochastic Hard attention and deterministic Soft attention. While generating the corresponding words of the output sentence for the given image (i.e., the caption of the given image), the researchers have managed to display the procedure involved for the model to automatically learn and fix its focus(attention) on the relevant objects and generates .[2]

The researchers claim that the model is quite accurate which has been verified qualitatively and quantitatively. For quantitative evaluation of the model, the researchers have calculated BLEU scores (BLEU-1, BLEU-2, BLEU-3 and BLEU-4), a metric used in machine translation to evaluate the quality of generated sentences, and METEOR scores for different data sets (Flickr8K, Flickr30K and MSCOCO) which shows the robustness of the model. This model aims to achieve significantly better performance in comparison to the current state of art approaches on all the three data sets. In terms of qualitative analysis, the model is more flexible and interpretive in terms of attention component that focuses on "non-object" relevant regions for visualization in comparison to other systems, which are based on object detection systems.

The BLEU-1 score achieved for Flickr8K data set on both Soft-Attention and Hard-Attention model is 67 as compared to BLEU-1 score of 65.6 for current state-of-the art. Similarly, on Flickr30K data set, the BLEU-1 score recorded was 66.7 and 66.9 for Soft-Attention and Hard-Attention respectively as compared to 60 for current state-of-the-art and for COCO it got improved from 66.6 to 70.7 and 71.8 for Soft-Attention and Hard-Attention model respectively.

The BLEU-2 score achieved for Flickr8K data set on both Soft-Attention and Hard-Attention model is 45.7 as compared to BLEU-2 score of 42.4 for current state-of-the-art. Similarly, on Flickr30K data set, the BLEU-2 score recorded was 43.4 and 43.9 for Soft-Attention and Hard-Attention model respectively as compared to 42.3 for current state-of-the-art and for COCO it got improved from 48.9 to 49.2 and 50.4 for Soft-Attention and Hard-Attention model respectively.

The BLEU-3 score achieved for Flickr8K data set on Soft-Attention model is 29.9 and Hard-Attention model is 31.4 as compared to BLEU-3 score of 27.7 for current sate-of-the-art. Similarly, on Flickr30K data set, the BLEU-2 score recorded was 43.4 and 43.9 for Soft-Attention and Hard-

[1] **Conference**: "The Journal of Machine Learning Research" in 2015, **Citations:** 3980, **Ranking:** A*

[2] **URL of paper**: https://arxiv.org/pdf/1502.03044.pdf

Attention model respectively as compared to 25.4 for current state-of-the-art and for COCO it got improved from 32.9 to 34.4 and 35.7 for Soft-Attention and Hard-Attention model respectively.

The BLEU-4 score achieved for Flickr8K data set on Soft-Attention model is 19.5 and Hard-Attention model is 21.3 as compared to BLEU-4 score of 17.7 for current sate-of-the-art. Similarly, on Flickr30K data set, the BLEU-4 score recorded was 19.1 and 19.9 for Soft-Attention and Hard-Attention model respectively as compared to 18.3 for current state-of-the-art and for COCO there was not much improvement for Soft-Attention model, while for Hard-Attention model there was some improvement with the score of 25.

The METEOR score achieved for Flickr8K data set on Soft-Attention model is 18.93 and Hard-Attention model is 20.3 as compared to METEOR score of 17.31 for current state-of-the-art. Similarly, on Flickr30K data set, the METEOR score recorded was 18.49 and 18.46 for Soft-Attention and Hard-Attention model respectively as compared to 16.88 for current state-of-the-art and for COCO it got improved from 20.71 to 23.90 and 23.04 for Soft-Attention and Hard-Attention model respectively.

II. Description of Original Data set

The authors have used 3 data sets: Flickr8K, Flickr30K and MS-COCO with 8000, 30000 and 82783 images respectively. Both Flickr8K and Flickr30K data sets consists of five (5) reference sentences for each image but for MS-COCO data set there were images which had more than 5 sentences describing the image which had been handled by basic tokenization to maintain the consistency of 5 sentences per image. The images are in ”jpeg” format and the reference sentences describing these images are stored in ”json” format.

III. Replication of original work

The repository on which we were working[3] is a Community version where the original authors code is reproduced for MS COCO data set. This includes downloading the data set, pre-processing, training the model, prediction and evaluation using the BLEU scores.

A. Download the data-set

A script provided by author which downloads the following:

1) Captions for validation and train images
2) ImageNet, VGG-19
3) Training images
4) Validation images
5) Unzips all of them, places the images in sub-directory (”image”) and captions in sub-directory (”data”).

B. Pre-processing

This involves res-sizing the file to 224*224 without changing the aspect ratio and then the re-sized images are saved in the images sub-directory. The prepro.py file build a list of dictionaries which contains captions, file-name and image ID. This implementation removes the caption with length more than 15. After extracting the features from the images, we save it to a pixel file.

We executed the code on Google Colab but faced memory related issues as the usage of RAM exceeded 25GB while running the model. To resolve this issue, we tried running the code on Amazon SageMaker (a fully managed service that provides the ability to build, train, and deploy machine learning models quickly) for which we placed a request for increase of RAM size from default 64GB to 256GB to the AWS team.

We had also setup GCP (Google cloud platform) notebooks instance for running the model as we faced issues with SageMaker resource limitations and Colab session time-outs. GCP had speed related issues as compared to its counterparts which did not allow to download the entire dataset of 13GB.

Training of the model took more than 12 hours on optimised TensorFlow because there was no pre-trained model available. There were lot of challenges while training the model:

1) Session limit time of 12 hours (We wrote a python script to overcome this problem)
2) There are different TensorFlow versions available, which caused lot of problem (The speed of the server depended on the TensorFlow version).

[3]**Github repository**: https://github.com/yunjey/ **s**how-attend-and-tell

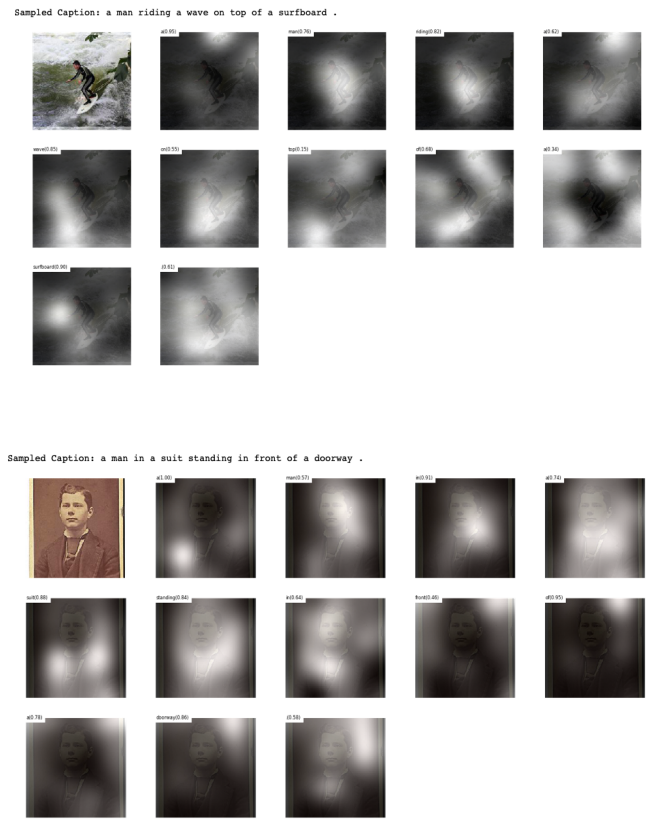

Fig. 1: *Attention maps generated by the model during evaluation*

On the other hand, we were also able to download the data set, train the model and perform prediction for new images on the cited data set (MS COCO) but could not obtain the evaluation BLEU scores for comparing and verifying authors claims through the code found on Tensorflow tutorials under image-captioning.[4]

IV. Construction of New Data Set

We have used MS COCO (100k) data set[5] as our new data set because creating a new data set for this project would be a challenge. The Microsoft Common Objects in COntext is a large-scale data set that addresses core research problems in scene understanding such as detecting non-iconic views (or non-canonical perspectives) of objects, contextual reasoning between objects and the precise 2D localization of objects. It is a data set with the goal of advancing the state-of-art in object recognition by placing the question of object recognition in the context of the broader question of scene understanding.

[4]**Author's code:** https://github.com/kelvinxu/arctic-captions

[5]**Data set link**: http://cocodataset.org/home

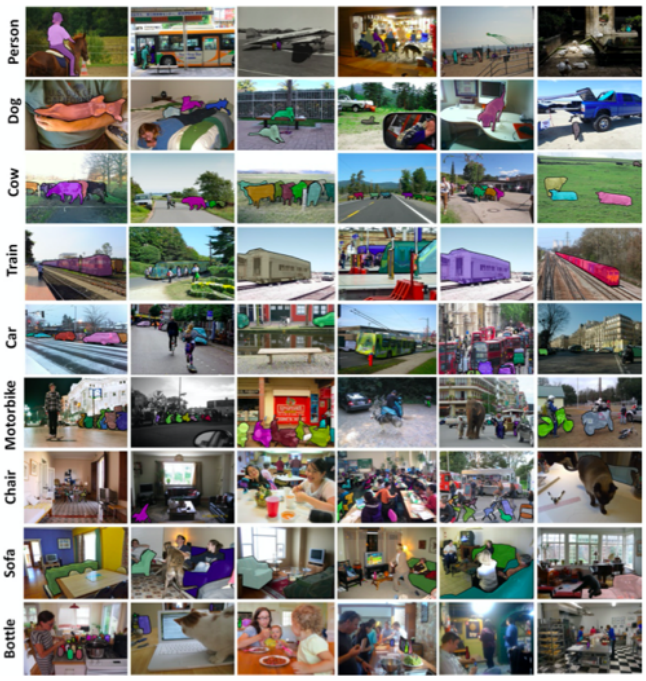

Fig. 2: *MSCOCO Dataset*

It has several features:

1) Object segmentation
2) Recognition in context
3) Super-pixel stuff segmentation
4) 330K images (>200K labelled)
5) 1.5 million object instances
6) 80 object categories
7) 91 stuff categories
8) 5 captions per image
9) 250,000 people with key points

The data set contains 91 common objects categories with 82 of them having more than 5000 labelled instances as shown below. In total the data set has 2,500,000 labelled instances in 328,000 images. In contrast to ImageNet data set, COCO has fewer categories but more instances per category. It has significantly larger number of instances per category than the PASCAL VOC and SUN data sets.

DATA SET STATISTICS: The properties of MS COCO data set in comparison to several other data sets is described below: Some of the other data sets comprise of ImageNet, PASCAL VOC

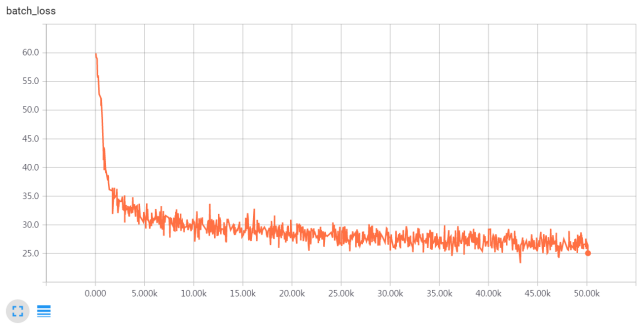

Fig. 3: *Loss curve*

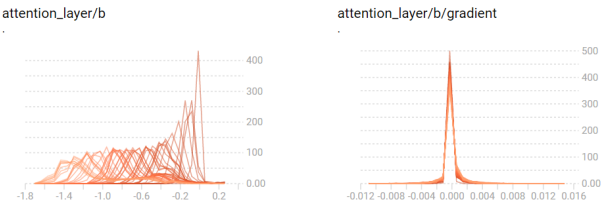

Fig. 4: *Attention layers and gradient plot*

2012, and SUN which vary significantly in size, list of labelled categories and types of images. ImageNet was created to capture a large number of object categories, many of which are fine-grained. SUN focuses on labelling scene types and the objects that commonly occur in them. Finally, PASCAL VOC 2012 focuses on object detection in natural images. The MS COCO is designed for the detection and segmentation of objects occurring in their natural context. It strives to find non-iconic images containing objects in their natural context. The COCO data set consists of 3.5 categories and 7.7 instances per image on average whereas data sets ImageNet and PASCAL VOC have less than two categories and three instances per image on average. Since the MS COCO data set does not consist of complicated (non-iconic) images of objects, models that are trained on MS COCO can generalize better than using data sets like PASCAL VOC.

V. Results on new data

We have considered the MS COCO data set (most challenging data set with 82,783 images) for our replication as constructing a new data set is a challenge. As being a quantitative evaluation, the results are reported with the frequently used BLEU $(metric)^2$, a standard in the caption generation

Dataset	Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR
Flickr8k	Google NIC	63	41	27	-	-
	Soft Attention	67	44.8	29.9	19.5	18.93
	Hard Attention	**67**	**45.7**	**31.4**	**21.3**	**20.30**
Flickr30k	Google NIC	66.3	42.3	27.7	18.3	-
	Soft Attention	66.7	43.4	28.8	19.1	**18.49**
	Hard Attention	**66.9**	**43.9**	**29.6**	**19.9**	18.46
MS COCO	Google NIC	66.6	46.1	32.9	24.6	-
	Soft Attention	70.7	49.2	34.4	24.3	**23.90**
	Hard Attention	**71.8**	**50.4**	**35.7**	**25.0**	23.04
OUR RESULTS						
MS COCO	**Validation set**	61.82	40.05	26.17	17.49	19.93
	Test set	61.42	39.56	25.75	17.11	19.81

Fig. 5: *BLEU-1,2,3,4/METEOR metrics compared to author's results*

literature. The BLEU scores from 1-4 are recorded. We have used MS COCO data set to run the code through the github repository (community version) and obtained the following scores.[6]

For evaluation, an iterative optimization algorithm (epochs) used in machine learning is used where we pass the full data set multiple times to the same neural network to optimise the learning. For our data set 20 epochs are used and results for the 20^{th} model is recorded.

From Fig 5. we observe that the BLEU scores we obtained are slightly less than author's scores which hereby, confirms successful replication of the paper.

- **Human Evaluation:**
  We have also used "Human Evaluation" to test the accuracy of model (how well the captions are generated by the model for a particular image in comparison to human annotation) with the classification accuracy (BLEU scores) obtained through the model. Here, we have randomly chosen 52 images from the MS COCO data-set, prepared an online google form [7] containing the images with given two options of captions. The first option is the caption where human describes the image and the second one is generated by the model. This survey has been circulated amongst five random people (responses) and the results are recorded and analysed. Through these results we have generated graphs for displaying how many responses in common have chosen captions generated by model or human annotation. The graphs also displays whether

[6]**Our Github repository**:https://github.com/sedhasukhdeep/show-attend-and-tell

[7]**Google form**:http://bit.ly/ADSform1

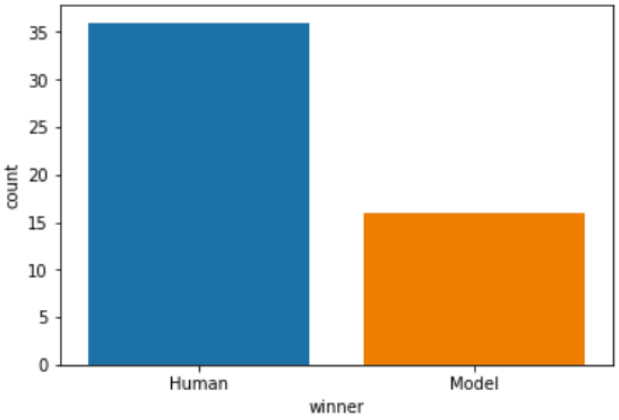

Fig. 6: *Graph displaying Agreement of five users on captions*

frequency of captions generated by model is higher than human annotations or vica-versa. Here, we have compared the user agreement for better quality of captions by taking a majority vote and also found Cohen Kappa's score of the responses (for reference look to our Github repo). According to our analyses we have found that percentage of responses who chose captions generated by the model is approximately half of the responses who chose human annotated captions.

VI. Reflection

There were lots of challenges faced while trying to implement both of the papers: "Show and Tell: A Neural Image Caption Generator" and "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention". The details regarding the challenges are mentioned below:

- **Obsolete libraries:** There were some libraries used by the authors while implementing their code which were obsolete, so we had hard time fixing these issues and handling the obsolete libraries in the code.
- **Dependency issues:** The code used to implement the paper were mostly written in Python 2.7 version so this created a huge challenge for us to implement the code in Python 3 version as there were lot of dependencies in different versions of Python. Also, the version of Tensorflow been used by the authors and one being currently used were different which also created hurdle for us to run the code. There were many other libraries used in the code which were updated recently and the updated version of libraries were not compatible with the existing code and were throwing exception.
- **Challenges in Google Colab:** Google Colab was one of the platform we chose to run the code. The main challenge we faced in Google Colab was that the session would automatically end if the system was idle for long time and when the session ended all the data was lost. As we were working with huge data set, this auto session time out issue compelled us to download the data set again and again which was cumbersome. Also, Google Colab had space issue and limited amount of computational resources. The usage of RAM exceeded 25GB while running the model which couldn't be handled by Google Colab due to memory space issue.
- **Challenges in AWS:** The main challenge we faced in AWS was the working environment provided by EC2 instance of AWS. As we were working with EC2 instance, we had access to command line only and had no GUI setup. We were dealing with images, and evaluating the output of the images in command line would have made the task complicated. So, in order to view the output of the images we had to first deal with the GUI setup in AWS EC2 instance which would have helped us view the outputs properly and easily. Another huge challenge faced in AWS was dealing with EC2 instance setup which required to specify all the details of that EC2 instance at the beginning and didn't have permission to modify it later as per required. We had to recreate the instances multiple times as we faced space issue, capacity issue, resources issues and libraries issues later while running the code. Creating a new instance again and again and beginning from the very first step was cumbersome.
- **Challenges in Sagemaker:** The main challenge to run the project (Show, Attend and Tell) using Sagemaker was the limited space resource available easily in Amazon Sage-

maker. The space required for the project was more than 64GB of RAM which was not available directly in Amazon Sagemaker, so we had to request the AWS team to provide us with the instance as required by the project.

- **Challenge in GCP(Google Cloud Platform):** The main challenge we faced in GCP was getting the right notebook instance which was problematic as while choosing the server configuration to run the instance, the error message was displayed at the end stating only certain number of CPUs, RAM and GPUs could be selected for the provided region which made the task redundant as none of the changes were saved. Withal to that, the other challenge we came across was the data transfer speed which was very slow. As the connection was poor, downloading the training data set, MSCOCO which is 13 GB in size, failed more than 7 times.
- **Size of data set (MSCOCO):** As we were dealing with huge data set, MSCOCO which is 13GB in size, downloading it in different environments and different platforms were time consuming and troublesome.

As mentioned earlier, we had tried different versions of codes for 2 papers: "Show and Tell: A Neural Image Caption Generator" and "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention". The overall generic challenges has been mentioned above and challenges specific to various repositories we worked on are mentioned below:

- **Show and Tell: Neural Talk I:** Neural Talk I was written in Tensorflow which is quite inefficient and old as training was very slow compared to other counterparts. The main challenge faced in this version was that the code doesn't work out of the box for new data. The code for extracting the raw features from each image was written in Matlab so it was complicated as first we had to pipe the images through VGG CNN to get the 4096-D activations on top which isn't that easy.
- **Show and Tell: Neural Talk II (Tensorflow version):** The main challenge in this version was that there were lot of dependencies in libraries and version of python and tensorflow and couldn't run the preprocessing part of the code.
- **Show and Tell: Neural Talk II (Docker version):** The main challege we faced in docker version was that we had to setup web interface to connect to the AWS virtual environment to install the docker. We failed to run the evaluation part in the docker due to python version issues which couldn't be resolved.
- **Show, Attend and Tell: (Author's code):** The challenge faced in this version of code was that the documentation of the readme file that instructed how to run the code was very poor and unclear so we couldn't run this version of code.
- **Show, Attend and Tell: (community version of Yunjey):** The challenge faced in this version of code was the requirement of a very large instance to run the code (RAM $>$ 64GB), also there was a requirement of minor changes in the training and evaluation files to get the desired output. This problem was overcomed by AWS SageMaker which ensured model kept on running with the large RAM.
- **Show, Attend and Tell: (community version of tensorflow):** The main challenge in this version of code was unavailability of the evaluation code which calculated BLEU scores.

VII. References - Github repositories

- https://github.com/karpathy/neuraltalk
- https://github.com/jazzsaxmafia/showandtell.tensorflow
- https://github.com/ruotianluo/neuraltalk2-tensorflow
- https://github.com/SaMnCo/docker-neuraltalk2
- https://github.com/jacopofar/neuraltalk2-web
- https://github.com/karpathy/neuraltalk2
- https://github.com/coldmanck/show-attend-and-tell
- https://www.tensorflow.org/tutorials/text/imagecaptioning
- https://github.com/kelvinxu/arctic-captions
- https://github.com/tylin/coco-caption
- https://github.com/yunjey/show-attend-and-tell