An Extensive Study on Image Captioning and proposal for new architecture*

Kalra Utkarsh
Data Science 4th Year Bachelor
Innopolis University
Innopolis, Russia

Kochekov Kerim
Data Science 4th Year Bachelor
Innopolis University
Innopolis, Russia

***Abstract*—Image captioning is a process in which an image is described textually referencing to different objects or aspects found in the image using Machine learning. There are a number of implementations available online involving the use of varied architectures of deep learning. In this project we aim to implement different proposed models, compare them and also propose some changes in the architecture to better the performance. Code has been made available at: https://github.com/kalraUtkarsh/Image_Captioning_CV**

***Index Terms*—NLP, Image-captioning, Semantic-Embedding, Deep-learning**

I. Introduction

Image Captioning is a problem which generally involves two main aspects of the Machine Learning/Deep Learning environment, that is dealing with images with Computer Vision and dealing with language interpretation with Natural Language Processing.
There are 4 main architectures available online to tackle this problem. Which will be discussed during the course of this project report. They are being presented and given background idea of each architecture separately below. In our extensive study of image captioning we came across an underlying problem faced by these kinds of projects which is selection of a Dataset, data processing and training, which in the course of the coming discussion, this problem is detailed and talked upon with proposals to better solutions and perspectives. **FLICKR 8K** and **MS-COCO** [15] Dataset is being used for the experiments across this project.
The methods we opted for this project are as follows:

- Understanding of the problem thoroughly with the help of different academic articles.
- Understanding the datasets.
- Finding state of the art implementations available online and analyzing them.
- Analyzing the loss functions and the metrics used and trying to replace them.
- Implementing the models from the architectures found.
- Trying out new architectures inspired by some pre-existent architectures.

There were also tremendous number of complications faced by us in terms of computational power and resources available to us, which is discussed in detail below.

II. Related Work

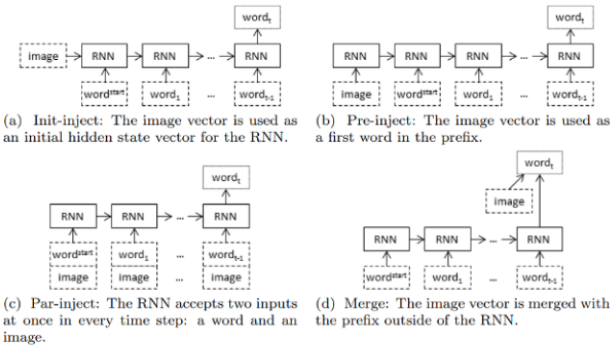
This particular problem does have a small amount of variety of solutions available on the internet and a handful of those are being discussed here in brief.

A. Main types of solutions proposed

Though across the vast span of internet there were quite a number of interesting solutions to be found but there are **4** main techniques or structures followed by all essentially.
Those being:

- By Marc Tanti and Albert Gatt.
- By Andrej Karpathy.
- Show and Tell: A Neural Image Caption Generator.
- Rich Image Captioning in the Wild

B. Marc Tanti and Albert Gatt Method



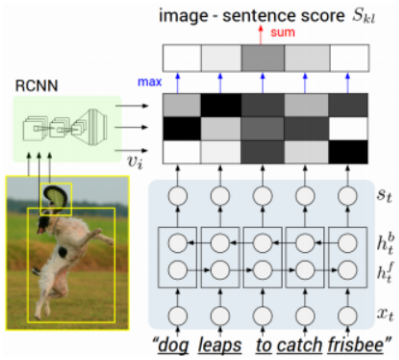
(a) Init-inject: The image vector is used as an initial hidden state vector for the RNN.

(b) Pre-inject: The image vector is used as a first word in the prefix.

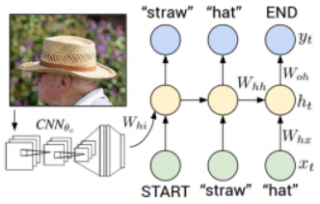
(c) Par-inject: The RNN accepts two inputs at once in every time step: a word and an image.

(d) Merge: The image vector is merged with the prefix outside of the RNN.

C. Andrej Karpathy Method

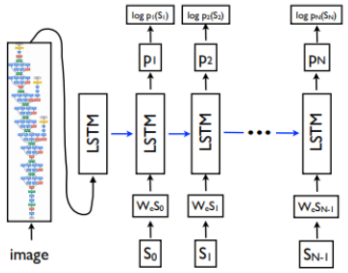


Object regions and CNN to recognize objects. For language modeling, the paper suggests the use of Bidirectional Neural Networks as it best represents the inter-modal relationships among the n-grams of the sentences.



The second part deals with the development of a Multimodal Recurrent Neural Network for generating captions.

D. Show and Tell: A Neural Image Caption Generator. Method



For obtaining the best prediction for the captions, the Architecture uses a Beam Search method.

E. Rich Image Captioning in the Wild Method

"Rich Image Captioning in the Wild" paper [4] proposed by Microsoft that was inspired by machine translation's encoder-decoder framework architecture. Architectures follows:
1. A deep Resnet based model for image feature extraction
2. A language model for caption candidate generation and ranking
3. An entity recognition for landmark and celebrities
4. A classifier to estimate the confidence score

Figure 2: Illustration of our image caption pipeline.

III. Methods

This section covers the methods and the steps done to realise the project goals.

A. Problem statement and explanation

The very first thing that was done was research about the topic and about the related work already available. For this we referred to a number of academic and non academic article to read about what is image captioning, how important this is and how much work is being done on it. The conclusion of the above research came out that though it can very useful in a number of feilds it is not yet being pushed in production in real life in a lot of places where it can be useful such as some security areas, the reasons being though it is not a far fetched or very rarely discussed topic, not a proper extensive amounts of research is done on it.
For this particular type of problem customarily used architecture are encoder decoder architectures which naturally involves an encoder and a decoder.
There are usually CNN layers/CNN models taking form of encoders as it retrieves the features of an image and then the last hidden state is connected to a decoder.
The decoder is usually a Recurrent Neural Network (RNN) which takes in the features from the CNN and models it upto word level.
Then takes place the Training and the testing part. Though the boiled down version of most models and architectures is coherently similar but at the same time they may differ quite a bit once we get into the nitty gritty of it.
We read and understood quite a few of those and selected ones to implement which we felt like were feasible given our limited resources and expertise.

B. Datasets

There are not a lot yet a considerable amount of datasets available for this particular problem. The main datasets are:

1) Common Objects in Context (COCO): COCO is a large-scale object detection, segmentation, and captioning dataset. It has over 330K images in over 80 object categories. Specifically for image captioning, each image has a series of *5 captions* associated to it for training and in some cases testing also. It is one of the largest and the most regarded dataset available for image captioning and segmentation. Yet for us this was one of the biggest problem to handle this data as the whole dataset spans over 20 Gigabites of data and does

require considerably large and robust resources.
An example from COCO:

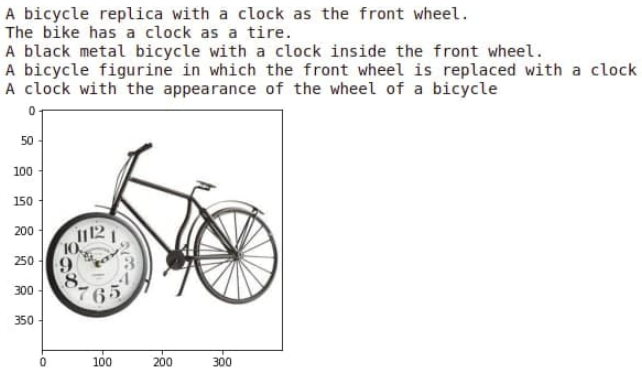


Fig. 1. An example from COCO

2) Flikr8k and Flikr30K: Similar to the COCO dataset discussed above this dataset also has a number of images spanning over different object categories and as the name itself suggests it has 8K images so it is a much smaller dataset, hence using Flikr8k was much more convenient for us. This dataset also has *5 dedicated captions* for each image instance. **Flikr30k** similar to its counterpart is a dataset with the standard structure but it requires a special license to access and has 30K images.
An example from Flikr8k:

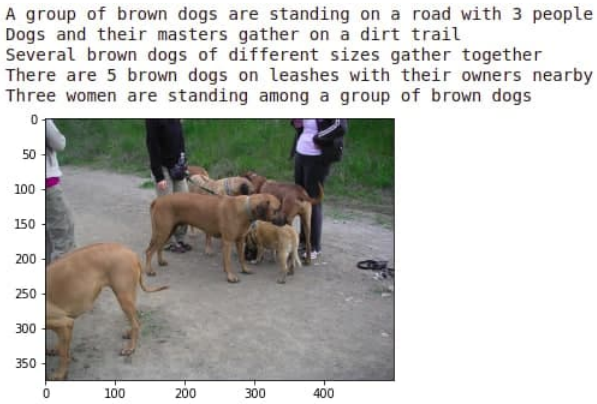


Fig. 2. An example from Flickr8k

3) Several 3rd party Datasets: There are several 3rd party datasets available online on sites like kaggle. But many of them are just some mismatches of the COCO or flikr dataset. Many of these datasets have mislabeled data and have less amount of images and captions associated with it.
We mainly analyzed COCO dataset and Flikr8k dataset properly and tried our test on them. For COCO we were unable to use the whole dataset so many-a-times we had to settle to using its validation dataset as our training dataset which was still humungous.

C. Analysing available solutions

As mentioned in the related works section. The main 4 techniques of the image captioning remains the same but some nitty gritty details are changed and experimented with. We found a number of implementations online most of which were in reference to online competitions for image captioning such as Google's Image captioning [14] and MS COCO image captioning competitions. The main and the biggest obstacle we faced in the whole project timeline was when we were trying to implement or try out these models so that we can understand how it works, the models were so complex and the datasets they used were so big that for us to even try out their solution was being too ambitious so we had to settle to reading their code and explanations.

D. Loss functions and Metrics used

This is one of the most complex and precarious parts of the whole image captioning project. There are many proposals online for loss functions but most of them present a problem or obstacle of some kind or another. As this problem and its result is highly subjective having a proper idea of right or wrong is very challenging and ambitious at the same time. In many cases available online people opted to use the loss from the loss function to be the metric itself.
The most commonly used loss function is the **crossentropy**, it works such as the output from the decoder is one-hot encoded arrays and to that we use the loss function cross entropy to train to model the get the desired one-hot encoded array.
One of the most commonly used Metric for image captioning we encountered was **BLEU**. BLEU stands for Bilingual Evaluation Understudy. Let's take an example to understand how BLEU works:
predicted caption= "the weather is good" **references:**

- the sky is clear
- the weather is extremely good

first, convert the predicted caption and references to unigram/bigrams.

$$\text{modified ngram precision} = \frac{max number of times ngrams occour in the refernece}{total number of ngrams in hypothesis}$$

The main problem as mentioned above with this kind of approach is, the captions may not have same words or similar but can make proper sense so it can be very much acceptable from a Human's perspective but will be having a very low score on this Metric. Many other metrics follow a very similar process.There is one Metric which tries to tackle this problem and does it somewhat successfully, proposed by Cornell University but it has some specific limitations which prevents it from being implemented easily.

E. Available Model Implementations

There were ways to test the models by just importing pretrained models and using them, but we did not find that

productive so we opted to just read the architecture and implement everything ourselves.

There were 2 main models that we were successful in implementing. The first model followed the following structure: use the VGG16 model that has been pre-trained for classifying images. But instead of using the last classification layer, we will redirect the output of the previous layer. This gives us a vector with 4096 elements that summarizes the image-contents - similar to how a ”thought-vector” summarized the contents of an input-text in Tutorial 22 on language translation[13]. We will use this vector as the initial state of the Gated Recurrent Units (GRU). However, the internal state-size of the GRU is only 512, so we need an intermediate fully-connected (dense) layer to map the vector with 4096 elements down to a vector with only 512 elements.The decoder then uses this initial-state together with a start-marker ”ssss” to begin producing output words. In the first iteration it will hopefully output the word ”big”. Then we input this word into the decoder and hopefully we get the word ”brown” out, and so on. Finally we have generated the text ”big brown bear sitting eeee” where ”eeee” marks the end of the text. The other model was of

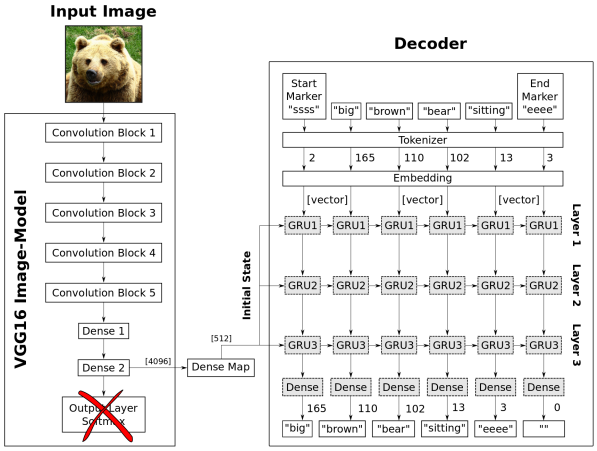


Fig. 3. Implemented 1st model

the structure as follows using pretrained model Resnet152[16] and LSTM memory cells: For the encoder part, the pretrained

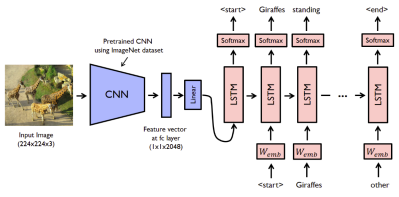


Fig. 4. Implemented 2nd model

CNN extracts the feature vector from a given input image. The feature vector is linearly transformed to have the same dimension as the input dimension of the LSTM network. For the decoder part, source and target texts are predefined. For example, if the image description is ”Giraffes standing next to each other”, the source sequence is a list containing [’¡start¿’, ’Giraffes’, ’standing’, ’next’, ’to’, ’each’, ’other’] and the target sequence is a list containing [’Giraffes’, ’standing’, ’next’, ’to’, ’each’, ’other’, ’¡end¿’]. Using these source and target sequences and the feature vector, the LSTM decoder is trained as a language model conditioned on the feature vector.

F. Our Model

When we refer to this project [17], which is merged version of some architectures we have discussed above, their model does not use images to train their model, they look images as just embedding representation which will help them to locate caption data. So this idea leads us to edit one of the architecture mentioned above(resnet151 + LSTM) to configure some hyperparameters, add some new players and most significantly add ”Dropout’ layer to CNN part to follow idea of not using images for training to get better and interesting results. Morevoer, try to train faster. As we refer to previous model it trained 3 epochs on ”MS-COCO 2014” data in 2 hours and 14 minutes, which is enourmous time. But with our new model it can be trained all 5 epochs in just 2 hours with better results. You can view results of provided some test images and loss function values over epochs below. More specifically here is our model’s used Encoder-Decoder architecture with specialities:

Max_answer_caption_length = 15

Encoder: Used Resnet50 (just deleting last layer to get FC 2048 sized layer) - Linear1 (2048, 512) - Dropout(p = 0.2) - Linear2 (512, 256)

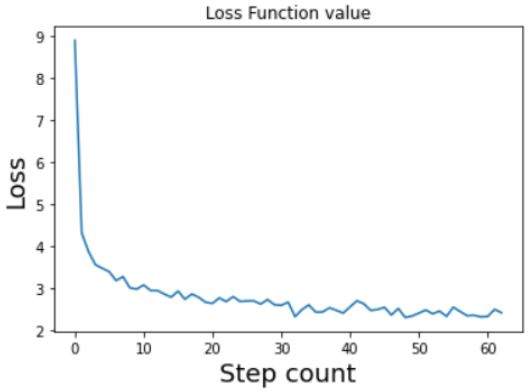
Decoder: Embedding(vocabulary_size, 256) - LSTM(256, 512, 2)

Strategy for predict(caption generate): Greedy-strategy (get maximum probability likelehood)

IV. Experiments and Evaluation

Here below you can see our test results for the images retrieved from the internet. Our model captioned them with interesting words by following English grammar structure well. We could also get another better results in continuing epochs, but as I mentioned above, it took near 40-45 minutes to train single epoch, so it is not possible to see that results under this circumstances. So here are some results:

Moreover, as we refer to this loss function chart, it gets converging as number of steps increasing, so we believe there is now meaning to waste our time to see slight better results.



V. Analysis and Observations

A. General Views and Analysis

All in all this was a project with a lot of intriguing and interesting aspects to it. We got to learn and implement a whole new alley of computer vision we were not aware of. The problem of Image captioning itself is part of bigger more robust problem sets out there yet a big chunk of it is dependent on human interpretations and a subjective facet of it, which intern makes it a very approachable yet very twisted problem at the same time. The whole idea at which this is based on is rather simple but ingenious as well. Our observations about the problem in general and the project are: That this is a much discussed problem yet a proper or state of the art solution to it is yet to come because as it has a lot of unmeasurable aspects to it, each solution has the potential to lack in one way or other. This is still a hot topic as big companies like google keep on having competitions on it time to time which intern shows the need, urgency and importance of this problem. It is a fun problem to work with and spend time on. There can be many other ways this problem can be solved and the potential is endless.

B. Problems Faced

As mentioned above in a few sections. We faced quite a number of big problems while executing our plans and trying to get some results. Biggest and main of them being lack of resources and time. We wanted to train the models on COCO dataset to get proper results or rather the best results as possible but we do not have enough Ram or internet or even space to get the dataset and use it. Naturally we could not train any models on or local machines as we don't have a GPU and big enough RAM so we had to turn to Colab but colab crashed all the time we had to start from the beginning each time which costed us a lot more time than expected. Even on Colab the training was taking too much time like more than 90 minutes for one epoch as the architectures are too complex.

VI. Conclusion

This paper showed how to generate captions for images. We used a pre-trained image-model VGG16 and ResNet151(and ResNet50) to generate a "thought-vector" of what the image contains, and then we trained a Recurrent Neural Network(LSTM, and GRU) to map this "thought-vector" to a sequence of words. This works reasonably well, although it is easy to find examples both in the training and validation-sets where the captions are incorrect.

It is also important to understand that this model doesn't have a human-like understanding of what the images contain. The model is merely a clever way of mapping pixels in an image to a vector of floating-point numbers that summarize the contents of the image, and then map these numbers to a sequence of integers-tokens representing words. So the model is basically just a very advanced function approximatory rather than human-like intelligence.

In the technical part the model can be improved in the future with Transformers, more specifically with diligent Attentions [18] to get better and interesting results. Meanwhile, for getting caption in the end the strategy can be changed to "Beam Search" rather than "Greedy search" to choose correct continuing vocabulary word with prefix embeddings.

All in all a very interesting project yet we feel like if we are given bit more robust and better resources and more time we can produce better results and try more models.

REFERENCES

[1] https://arxiv.org/pdf/1703.09137.pdf
[2] https://cs.stanford.edu/people/karpathy/cvpr2015.pdf
[3] https://arxiv.org/pdf/1411.4555.pdf
[4] https://papers.nips.cc/paper/2019/file/680390c55bbd9ce416d1d69a9ab4760d-Paper.pdf
[5] https://towardsdatascience.com/a-guide-to-image-captioning-e9fd5517f350
[6] https://towardsdatascience.com/bleu-bilingual-evaluation-understudy-2b4eab9bcfd1
[7] https://medium.com/@raman.shinde15/image-captioning-with-flickr8k-dataset-bleu-4bcba0b52926
[8] https://towardsdatascience.com/image-captioning-in-deep-learning-9cd23fb4d8d2
[9] https://towardsdatascience.com/foundations-of-nlp-explained-bleu-score-and-wer-metrics-1a5ba06d812b
[10] https://www.kaggle.com/adityajn105/flickr8k
[11] https://vision.cornell.edu/se3/wp-content/uploads/2018/03/1501.pdf
[12] https://openaccess.thecvf.com/content_CVPR_2020/html/Cornia_Meshed-Memory_Transformer_for_Image_Captioning_CVPR_2020_paper.html
[13] https://github.com/Hvass-Labs/TensorFlow-Tutorials/blob/master/22_Image_Captioning.ipynb
[14] https://ai.google.com/research/ConceptualCaption
[15] https://cocodataset.org/
[16] https://arxiv.org/abs/1512.03385
[17] https://github.com/abr-98/Image$_{C}aptioning_flickr$
[18] http://papers.nips.cc/paper/7181-attention-is-all-you-