

DRishti

Image Description Project To aid the visually impaired

Minor Project report

Submitted by:

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 16103143 | Shaurya Pratap Singh | B6 |
| 16103293 | Adarsh Singh | B6 |
| 16103239 | Tanya Dwivedi | B6 |
| 16103104 | Garima Malhotra | B3 |

Submitted to: Dr. Archana Purwar

**CERTIFICATE**

This is to certify that this group of batch B6-B3 has completed the minor project entitled “DRISHTI: Image Description Project to aid the visually impaired” themselves and under my guidance and to my complete satisfaction. The process of the project had been continuously reported and they have been in my acknowledgement constantly.

**Dr. Archana Purwar**

**Acknowledgment**

We would like to extend our profound sense of gratitude towards Dr. Archana Purwar who gave us this golden opportunity to embark ourselves on this struggling yet enjoyable journey of working on this project. This project consumed exorbitant amount of hard work, dedication and research but still the implementation would not have been possible without her meticulous and utilitarian guidance, who also provided us with all the much required resources and study material and guided us all through the project. Many people, especially our classmates and team members itself, have made valuable comment suggestions on this proposal which gave us an inspiration to improve our assignment. We thank all the people for their help directly and indirectly to complete our assignment.

**Table Of Contents**

1. Introduction
2. Problem Statement
3. Sustainable Development Goals
4. Technologies and Tools Used
5. Overview
6. Previous Accomplishments
7. Our Contributions
8. DataSet
9. FlowChart
10. Working
11. Snapshots
12. Research Paper Report
13. Accuracy
14. Future Scope
15. References
16. Conclusion

**Introduction**

**As the standards of “Intelligence” rise among the human beings, the machines are no less in improving their “Artificial Intelligence” to aid the humans in all the possible ways.**

**Image description** is the process of generating textual description of an image. It uses both **Natural Language Processing** and **Computer Vision** to generate the captions.

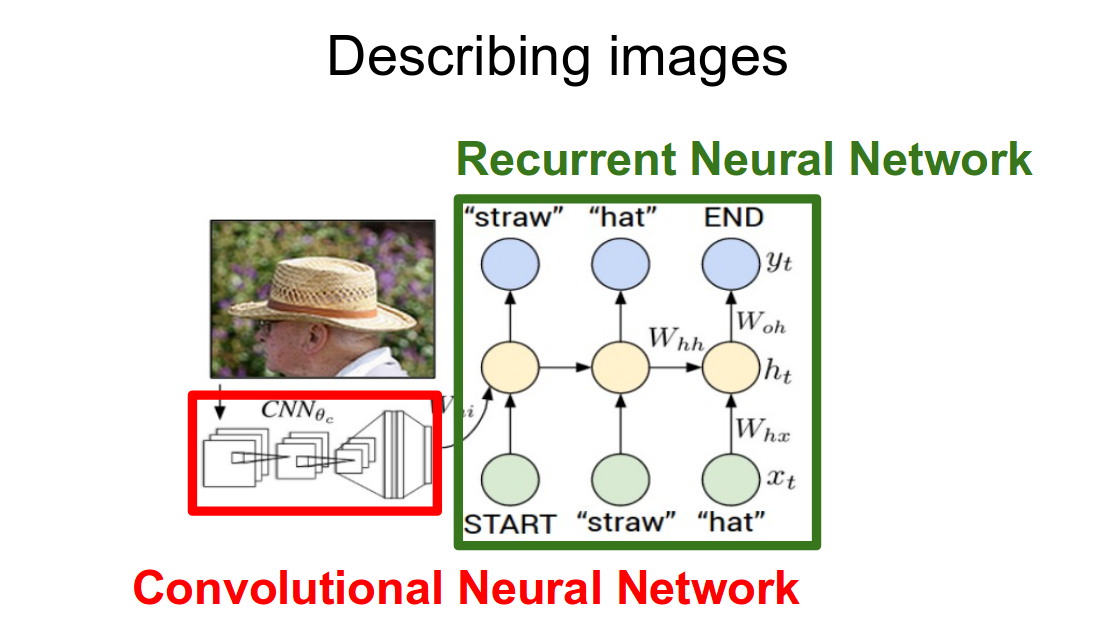
Image caption generation has emerged as a challenging and important research area following advances in statistical language modeling and image recognition. The generation of captions from images has various practical benefits, ranging from aiding the visually impaired, to enabling the automatic and cost-saving labeling of the millions of images uploaded to the Internet every day.

The field also brings together state-of-the-art models in **Natural Language Processing** and **Computer Vision**, two of the major fields of Artificial Intelligence.



**Problem Statement**

**Consider an image…**



**What is the first thing that comes to your mind when you see this image???**

1. **A man wearing a hat.**
2. **An old man wearing a straw hat.**
3. **An old man wearing glasses and a straw hat standing near trees.**

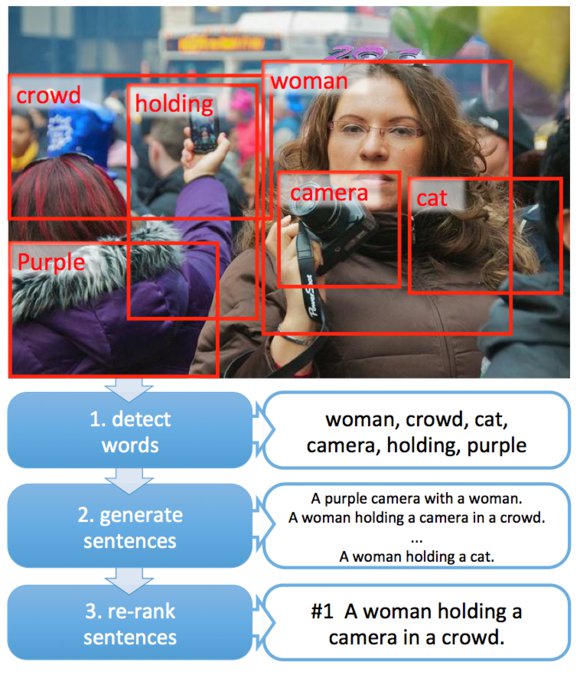
**A quick glance on the image is sufficient to understand and describe.**

**Automatically generating this textual description from an artificial system is the task of image description.**

The task is straightforward – the generated output is expected to describe in a single sentence what is shown in the image – the objects present, their properties, the actions being performed and the interaction between the objects, etc. But to replicate this behaviour in an artificial system is a huge task, as with any other image processing problem and hence the use of complex and advanced techniques such as Deep Learning to solve the task.

Captioning an image involves generating a human readable textual description given an image, such as a photograph. The problem is easy very for a human brain to process but for a machine it becomes very difficult to describe the image from its pixels.

This might seem a very simple problem for human eye, however it is a very hard task for a machine to interpret the image.



**Sustainable Development**

Being able to automatically describe the content of an image using properly formed English sentences is a challenging task, but it could have great impact by helping visually impaired people better understand their surroundings. The user can capture photographs of their environments. These images can then be used to generate captions that can be read out loud to the **visually impaired**, so that they can get a better sense of what is happening around them.

**Motivation**

Interpreting the world for the people who are blind or visually impaired. APIs from Microsoft Cognitive Services with the imaging performance and power of Pivothead SMART, a person who is visually-impaired can better understand who and what is going on around them. While wearing the glasses, the user swipes the touch panel on the eyewear to take a photo. The eyewear will analyze and translate the image to speech and describe what the person is doing, how old they are, and what emotion they're expressing. A user can take an image of text - from a nutrition label to a news article - and the eyewear will read it to the user.

“It’s about taking the power of human language and applying it more pervasively to all of our computing.”

— Satya Nadella, Microsoft CEO

**Technologies & Tools Used**

1. **Python**
2. **Dataset**

* **Flickr8k Dataset**

1. **Frameworks**

* **Tensor flow**
* **Keras**

1. **Machine Learning**

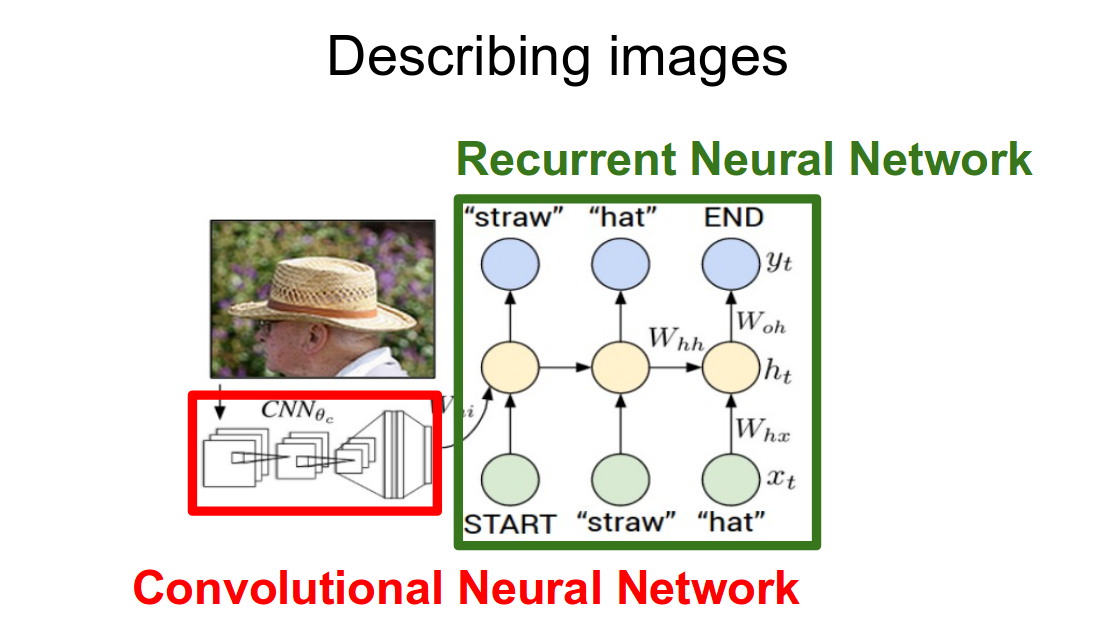
* **OpenCV**
* **Convolution Neural Network (CNN)**
* **Long Short Term Memory (LSTM)**
* **Recurrent Neural Network (RNN)**
* **Natural Language Processing (NLP)**

**Overview**

The task of image captioning can be divided into two modules logically – one is an **image based model** – which extracts the features and nuances out of our image, and the other is a **language based model** – which translates the features and objects given by our image based model to a natural sentence.

For our image based model (viz. encoder) – we usually rely on a Convolution Neural Network model.

And for our language based model (viz. decoder) – we rely on a Recurrent Neural Network. The image below summarizes the approach given above.



Usually, a pretrained CNN extracts the features from our input image. The feature vector is linearly transformed to have the same dimension as the input dimension of the RNN/LSTM network. This network is trained as a language model on our feature vector.

**Environment**

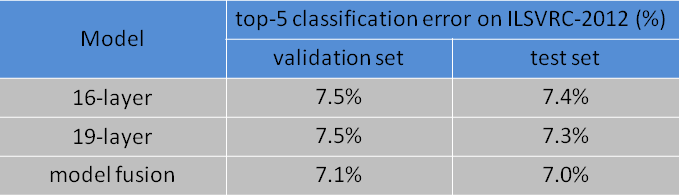
We have used Python SciPy environment, ideally with Python 3. Keras (2.1.5 or higher) with either the TensorFlow or Theano backend. We havescikit-learn, Pandas, NumPy, and Matplotlib installed.

## Photo and Caption Dataset

A good dataset to use when getting started with image captioning is the **Flickr8K** dataset. The reason is because it is realistic and relatively small so that we can download it and build models on our workstation using a CPU. The dataset has a pre-defined training dataset (6,000 images), development dataset (1,000 images), and test dataset (1,000 images).

## Preparation of Photo Data

We will use a pre-trained model to interpret the content of the photos. There are many models to choose from. In this case, we will use the Oxford Visual Geometry Group, or **VGG, model**.



Keras provides this pre-trained model directly. We could use this model as part of a broader image caption model. The problem is, it is a large model and running each photo through the network every time we want to test a new language model configuration (downstream) is redundant.

Instead, we can pre-compute the “photo features” using the pre-trained model and save them to file. We can then load these features later and feed them into our model as the interpretation of a given photo in the dataset. It is no different to running the photo through the full VGG model; it is just we will have done it once in advance. This is an optimization that will make training our models faster and consume less memory. We can load **the VGG model in Keras** using the **VGG class**. We will remove the last layer from the loaded model, as this is the model used to predict a classification for a photo. We are not interested in classifying images, but we are interested in the internal representation of the photo right before a classification is made. These are the “features” that the model has extracted from the photo. Keras also provides tools for reshaping the loaded photo into the preferred size for the model (e.g. 3 channel 224 x 224 pixel image).

**Preparation of text data:**

The dataset contains multiple descriptions for each photograph and the text of the descriptions requires some minimal cleaning. Each photo has a unique identifier. This identifier is used on the photo filename and in the text file of descriptions.  Each photo identifier maps to a list of one or more textual descriptions. We will clean the text in the following ways in order to reduce the size of the vocabulary of words we will need to work with:

* Convert all words to lowercase.
* Remove all punctuation.
* Remove all words that are one character or less in length (e.g. ‘a’).
* Remove all words with numbers in them.

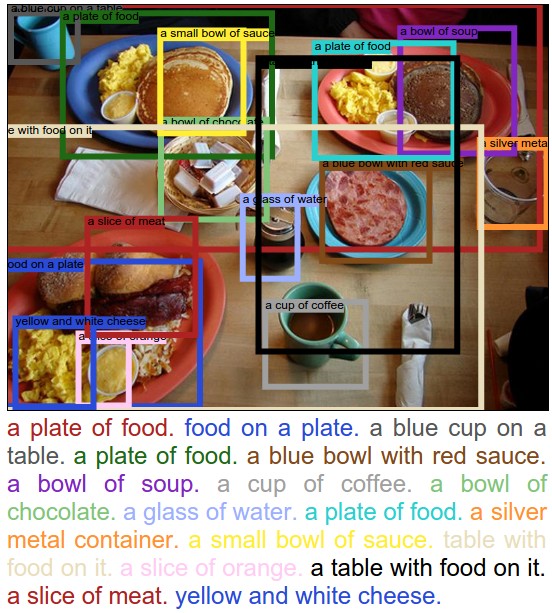
Once cleaned, we can summarize the size of the vocabulary. Ideally, we want a vocabulary that is both expressive and as small as possible. A smaller vocabulary will result in a smaller model that will train faster. For reference, we can transform the clean descriptions into a set and print its size to get an idea of the size of our dataset vocabulary.

**What is encoder?**

The neural networks that changes the any input in its features representation using vector of numbers is encoder. For example, we want to use image to predict words. As image directly can't tell what should be the word, we want to use its feature to help us decide the next word. And thus the network of layers used to change image or any other type of input in its feature representation is known as encoders.

**What is decoder?**

The combination of layers/neural network that takes feature representation provided by encoder as its own input and predicts the next word, is known as decoder.



**Previous Accomplishments**

The problem of generating natural language descriptions from visual data has long been studied in computer vision. This has led to complex systems composed of visual primitive recognizers combined with a structured formal language, e.g. And-Or Graphs or logic systems, which are further converted to natural language via rule-based systems. Such systems are heavily hand-designed, relatively brittle and have been demonstrated only on limited domains, e.g. traffic scenes or sports. The problem of still image description with natural text has gained interest more recently. In this work we combine deep convolutional nets for image classification with recurrent networks for sequence modeling, to create a single network that generates descriptions of images. The model is inspired by recent successes of sequence generation in machine translation, with the difference that instead of starting with a sentence, we provide an image processed by a convolutional net.

**Our Contributions**

We use a more powerful RNN model, and provide the visual input to the RNN model directly, which makes it possible for the RNN to keep track of the objects that have been explained by the text. As a result of these seemingly insignificant differences, our system achieves substantially better results on the established benchmarks.

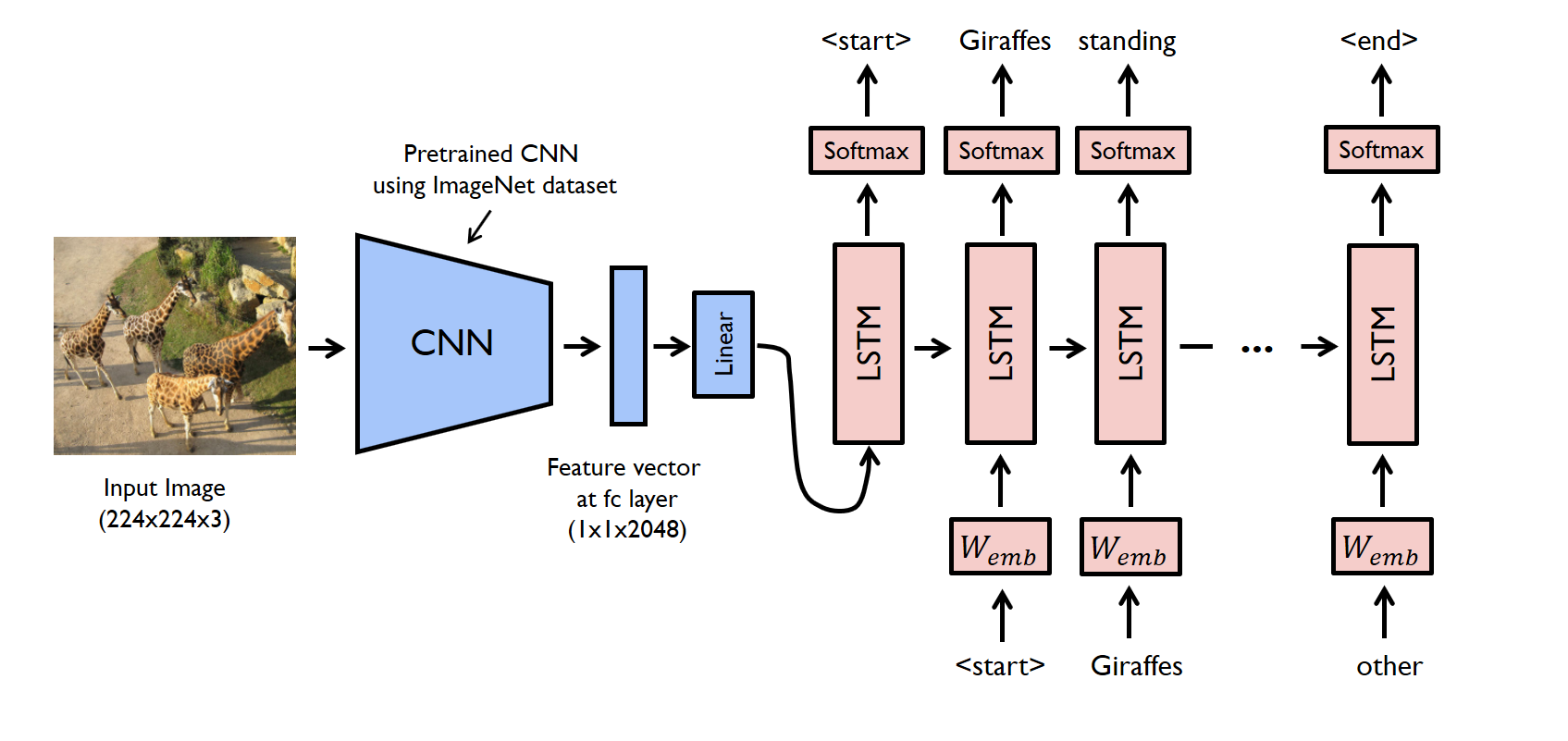
1. The model used for extraction of features is VGG16 model which is a Pre-Trained Photo Model to improve the feature extraction of the model.
2. To increase the accuracy progressively in predicting the captions.
3. To be able to generate captions for images taken in real time.
4. To improve the LSTM-based decoder model to reduce the search scope of the vocabulary for the sequence generation.
5. Most of the literature deals with models that generate image descriptions in the English language, emphasized by the fact that the descriptions used in training and for benchmarking are also in the English language.

Our project will be focused on developing models adapted to the generation of captions in other languages also using APIs.

**DataSet**

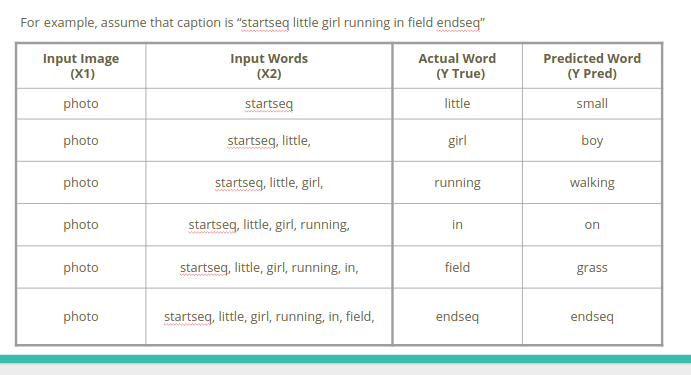
For this project we have used the Flickr8k dataset which contains 6,000 train images, 1000 test images and 1000 validation images that are each paired with five different captions which provide clear descriptions of the salient entities and events. The images were chosen from six different Flickr groups, and tend not to contain any well-known people or locations, but were manually selected to depict a variety of scenes and situations. Each image contains 4- 5 descriptions. The size of the dataset is nearly 1.1 GB.

**FlowChart**



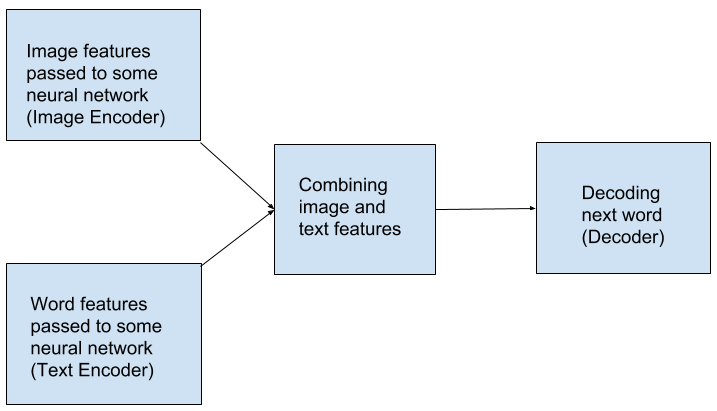
**Working**

1. First thing that comes into mind is that how are we going to restructure our dataset in order to train it.
2. We are going to generate one word at a time in order to generate complete sentence. To generate each word, we will provide 2 types of inputs.
   1. Image
   2. Sentence that has already been predicted so that model can use the context and predict next word.
3. How our training sample is going to look?
   1. We have to add 2 special tokens to each captions that represents start of sentence and end of sentence.
   2. Then we need to split each caption and image pair in multiple data samples.



1. So corresponding to a single image and a caption, we are going to generate multiple data samples.

**Structure of the model :**

****

### Extracting image features

1. To extract image features, we can use CNN network. Either we can make our own CNN network or we can use concept of transfer learning. There are a lot of pre-trained models available to extract image features. For example, VGG16 model contains 16 layers which is used to classify image in 1 out of 1000 classes. Last layer of this model is used for classification, so we can capture the output of second last layer (which is a vector of size 4096 for a single image) as it will represent image features in form of numbers without classifying them in classes.
2. So, we can pass each image in our dataset through this network and store the results corresponding to image id in a file.

# During training Each description will be split into words. The model will be provided one start word and the photo and generate the next word.

# Then the first two generated words of the description will be joined to form a sequence and provided to the model as input with the image to generate the next word. The model will stop when either gets the end word or any word appears which is out of our dictionary.

# While doing this, model will be keeping track of its performance as we also know the next actual word. This is how model learns that corresponding to the given image features and given input sequence, what the predicted output should be like.

During testing: When a test image is passed firstly we extract the features of the image by passing it into the VGG16 model. Then we feed the startseq (seed) to our model and the image features.

An important step during the process is to convert the predicted sequence into the integer encoded sequence and then pass into the model because a model can only interpret the language in numbers and not words, etc. The model predicts the words with certain probabilities, the index of word which has the highest probability is converted back to word with the help of word tokenizer and finally the predicted word is appended to our answer sequence and then this process is repeated till the endseq occurs or any word which is out of our vocabulary occurs. So we stop and return the answer sequence.

This is how our model works!

When a recurrent neural network language model is used for caption generation, the image information can be fed to the neural network either by directly incorporating it in the RNN -- conditioning the language model by `injecting' image features -- or in a layer following the RNN -- conditioning the language model by `merging' image features. We have defined a deep learning based on the **“**merge-model**”.** The merge architecture does have practical advantages, as conditioning by merging allows the RNN's hidden state vector to shrink in size by up to four times.

The Photo Feature Extractor model expects input photo features to be a vector of 4,096 elements. These are processed by a Dense layer to produce a 256 element representation of the photo. The Sequence Processor model expects input sequences with a pre-defined length (34 words) which are fed into an Embedding layer that uses a mask to ignore padded values. This is followed by an LSTM layer with 256 memory units.

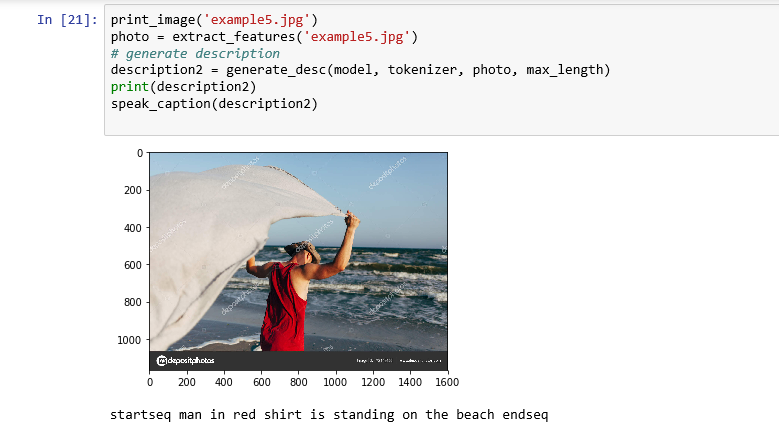
Both the input models produce a 256 element vector. Further, both input models use regularization in the form of **50% dropout**. This is to ***reduce overfitting*** the training dataset, as this model configuration learns very fast.The Decoder model merges the vectors from both input models using an addition operation. This is then fed to a Dense 256 neuron layer and then to a final output Dense layer that makes a softmax prediction over the entire output vocabulary for the next word in the sequence.

We also create a plot to visualize the structure of the network that better helps understand the two streams of input.

**Snapshots**

****

****

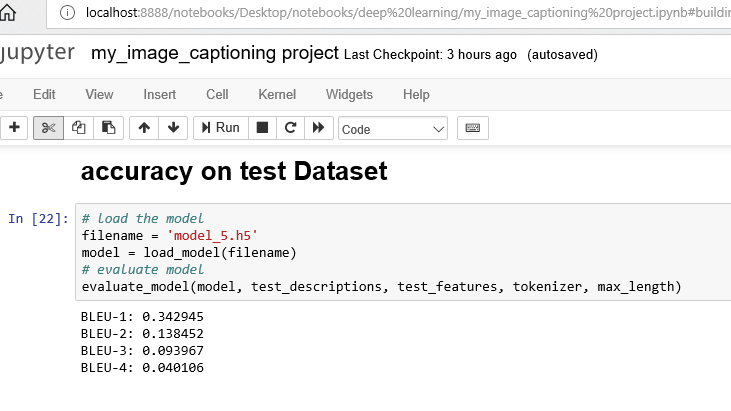
****

**Research Paper Report**

1. We use a more powerful RNN and LSTM model, and provide the visual input to the RNN model directly, which makes it possible for the RNN to keep track of the objects that have been explained by the text. As a result of these seemingly insignificant differences, our system  
   achieves substantially better results on the established  
   benchmarks.
2. The model used for extraction of features is VGG16 model which is a Pre-Trained Photo Model to improve the feature extraction of the model.
3. Moreover the use of VGG16 model makes our compilation of the model very fast and reduces the load on the system at the time of training.
4. Recent advances in machine translation have shown an approach that can be used to circumvent these 2 problems and solve the main problem directly. For many years, machine translation was also achieved by a series of separate tasks (translating words individually, aligning words, reordering, etc), but recent work has shown that translation can be done in a much simpler way using Re current Neural Networks (RNNs) and still reach state-of-the-art performance. An “encoder” RNN reads the source sentence and transforms it into a rich fixed-length vector representation, which in turn in used as the initial hidden state of a “decoder” RNN that generates the target sentence.

|  |  |
| --- | --- |
| In our project | In respective research paper |
| 1.Dataset: Flickr8k   * **Flickr8k\_images:**Contains 8092 photographs in JPEG format. * **Flickr8k\_text**: Contains a number of files containing different sources of descriptions for the photographs. | MOSCOCO,COCO  data comprises 82783 training images,  40504 validation images and 40775 test images. |
| 2.Evaluation of model: BLUE score   |  |  | | --- | --- | | 1  2  3  4 | BLEU-1: 0.579114  BLEU-2: 0.344856  BLEU-3: 0.252154  BLEU-4: 0.131446 | | BLUE and CIDEr score:  0.118 BLEU4 points and  48.9 CIDEr points |
| 3.Preparing photo data:  VGG16: very deep convolutional network. Keras provides this pre-trained model directly but the problem is we have to download it again and again, so Instead, we can pre-compute the “photo features” using the pre-trained model and save them to file. We can then load these features later and feed them into our model as the interpretation of a given photo in the dataset. | Convolutional neural networks:  Overfitting is the most problems of CNNs  High computational cost.  Need good GPU they are quite slow to train. |
| 4.Preparing text data:  We cannot feed raw text directly into deep learning models. Text data must be encoded as numbers to be used as input or output for machine learning and deep learning models. The first step in encoding the data is to create a consistent mapping from words to unique integer values. Keras provides theTokenizerclass that can learn this mapping from the loaded description data. | 1. words are represented as a one-hot vectors. 2. Hash Encoding with hashing\_trick. |
| 5.Main model: the image information can be fed to the neural network either by directly incorporating it in the RNN -- conditioning the language model by `injecting' image features -- or in a layer following the RNN -- conditioning the language model by `**merging**' image features. | Same as done in our project. |
| * 6.Model Description: * **Photo Feature Extractor**. This is a 16-layer VGG model pre-trained on the ImageNet dataset. We have pre-processed the photos with the VGG model (without the output layer) and will use the extracted features predicted by this model as input. * **Sequence Processor**. This is a word embedding layer for handling the text input, followed by a Long Short-Term Memory (LSTM) recurrent neural network layer. * **Decoder**:Both the feature extractor and sequence processor output a fixed-length vector. These are merged together and processed by a Dense layer to make a final prediction. | We use a deep convolutional neural network to generate a vectorized representation of an image that we then feed into a Long-Short-Term Memory (LSTM) network, which then generates captions. |
| 7.Reduction of overfitting: Both the input models produce a 256 element vector. Further, both input models use regularization in the form of **50% dropout**. This is to reduce overfitting the training dataset, as this model configuration learns very fast. | dropout keep probability = 0.75. |
| 8.Fitting the model:  When the skill of the model on the development dataset improves at the end of an epoch, we will save the whole model to file. At the end of the run, we can then use the saved model with the best skill on the training dataset as our final model. We can do this by defining a ModelCheckpoint in Keras and specifying it to monitor the minimum loss on the validation dataset and save the model to a file that has both the training and validation loss in the filename.  Loss is measured using the cross entropy function. | evaluated the parameters of the model at each iteration using the cross entropy loss of the predictions on each sentence. |
| 9.Removing memory error:  Using Progressive Loading, Generator Function and Instance Reduction technique. | The code they are running is on EC2 instancewith 32GB or 64GB of RAM. so there was no optimization of memory, but we have to run it on our laptops with 4GB or 8GB of RAM so we have optimized it. |

**Accuracy**

****

**It is quite appreciable bleu scores with respect to minimal the amount of train(6000) images used.**

**The scores could be further increased by using bigger Datasets such as COCO which contains 120000 set of images .**

**Moreover a good computational technology can boost up our results as we have only used just one description per image to train which could reduce some overfitting.**

Once the model is fit, we evaluated the skill of its predictions on the holdout test dataset. We evaluated a model by generating descriptions for all photos in the test dataset and evaluating those predictions with a standard cost function. First, we need to be able to generate a description for a photo using a trained model. This involves passing in the start description token **‘**startseq**‘**, generating one word, then calling the model recursively with generated words as input until the end of sequence token is reached **‘**endseq**‘** or the maximum description length is reached. We generated predictions for all photos in the test dataset and in the train dataset.  The actual and predicted descriptions are collected and evaluated collectively using the ***corpus BLEU score*** that summarizes how close the generated text is to the expected text. BLEU scores are used in text translation for evaluating translated text against one or more reference translations. Here, we have compared each generated description against all of the reference descriptions for the photograph. We then calculated BLEU scores for 1, 2, 3 and 4 cumulative n-grams. We get the following BLUE score: BLEU-1 0.579114, BLEU-2 0.344856, BLEU-3 0.252154, BLEU-4 0.131446.

**Future Scope**

Future work could focus on translating videos directly to sentences instead of generating captions of images. Static images can only provide blind people with information about one specific instant of time, while video caption generation could potentially provide blind people with continuous real time information. LSTMs could be used in combination with CNNs to translate videos to English descriptions.

An intelligent camera app can be made where you just hold up your phone and hear information about the world around you. It can speak short text as soon as it appears in front of the camera, provide audio guidance to capture a printed page, and recognizes and narrates the text along with its original formatting. The app can also scan barcodes with guided audio cues to identify products, recognize and describe people around you and their facial expressions, as well as describing scenes around you using the power of AI.

1. Turns the visual world into an audible experience — with this intelligent camera app, just hold up your phone and hear information about the world around you.
2. Recognize and locate the faces of people you’re with, as well as facial characteristics, approximate age, emotion, and more.
3. Read text quickly — hear short snippets of text instantly and get audio guidance to capture full documents.

**References**

**Research papers:**• Show and Tell: A Neural Image Caption Generator, 2015.  
• Show, Attend and Tell: Neural Image Caption Generation with  
Visual Attention, 2015.  
• Long-term recurrent convolutional networks for visual recognition  
and description, 2015.  
• Deep Visual-Semantic Alignments for Generating Image  
Descriptions, 2015.  
• ”Visual Impairment and Blindness.” World Health Organization.  
(2014). Web. 10 Apr. 2016  
• <https://www.analyticsvidhya.com/blog/2018/04/solving-an-image-captioning-task-using-deep-learning/>

**Articles:**• Automatic image annotation on Wikipedia  
• Long Short Term Memory RNN Wikipedia:  
 <https://en.wikipedia.org/wiki/Long_short-term_memory>.  
• Show and Tell: image captioning open sourced in TensorFlow,  
2016.  
• Medium articles  
• stack overflow

**YouTube videos:**

Siraj Raval

Andrew NG, Sent Dex etc.

**Conclusion**

The primary focus was on the distinction between what we have termed ‘inject’ and ‘merge’ architectures. Furthermore, by keeping language and image information separate, merge architectures lend themselves to potentially greater portability and ease of training. RNN decides on which word is the most likely to be generated next, given what has been generated before. In multimodal generation, this view encourages architectures where the image is incorporated into the RNN along with the words that were generated in order to allow the RNN to make visually informed predictions. The second view is that the RNN’s role is purely memory-based and is only there to encode the sequence of words that have been generated thus far. This representation informs caption prediction at a later layer of the network as a function of both the RNN encoding and perceptual features. Caption generation turns out to perform worse, in general, when image features are injected into the RNN. Thus, the role of the RNN is better conceived in terms of the learning of linguistic representations, to be used to inform later layers in the neural network, where predictions are made based on what has been generated in the past together with the image that is guiding the generation. We conducted an extensive hyper-parameter search over the CNN-LSTM model architecture, producing a best model that achieves results that are 0.131446 BLEU-4 points behind the state-of-the-art, using a keep probability of 50% for dropout and two layers for our decoder LSTM network. We demonstrated that semantically-close emitted words (e.g. ’plate’ and ’bowl’) result in similar movements in hidden state despite different previous context and that divergences in hidden state occur only upon emission of semantically-far words (e.g. ’vase’ and ’food’). We observed that there was not any major variation across the use of different CNNs. We could also see the limitations of the model based on the training dataset.

We have presented a deep learning model that automatically generates image captions with the goal of helping visually impaired people better understand their environments. Our described model is based on a CNN that encodes an image into a compact representation, followed by a RNN that generates corresponding sentences based on the learned image features.

We showed that this model achieves comparable to state-of-the-art performance, and that the generated captions are highly descriptive of the objects and scenes depicted on the images, because of the high quality of the generated image descriptions, visually impaired people can greatly benefit and get a better sense of their surroundings using text-to-speech technology.

****