PYTHON/DEEP LEARNING

PROJECT

ON

IMAGE CAPTIONING

Done by:

Naga Srividya Varanasi (Class ID: 22)

Chandra Sekhar Pentakota (Class ID: 27)

Pujitha Koppanathi (Class ID: 12)

**MOTIVATION**

Every day we come across many sources like social networking, news, advertisements or blogs where we encounter so many images. We have to interpret those images and most of these images do not have any description. Artificial Intelligence has may research areas like speech recognition, facial recognition, handwriting recognition and many others. So, image captioning is one among the research areas which mostly deals with the understanding and language description of the image. This aroused interest in us to do some challenging project which deals with understanding the scene type of the image, location and extracting the main features from the image and also generating the syntactically and semantically correct sentences.

Understanding images mainly deals with obtaining image features. This can be done using deep learning-based techniques. Through deep learning, the features are learnt automatically from the training data and by evaluating they can handle large and diverse set of images and videos. Also, through deep learning complex algorithms and challenges can be handles pretty well. The availability of large data sets has also led to the increase in the use of deep learning algorithms.

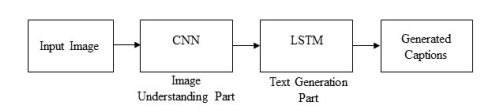
Most of the image captioning based models come under novel caption generation. They first look into the visual content of the image and then generate image captions form the visual content using a programming language.

**PROJECT GOAL**

The main aim of the project is to make the machine understand of the image and give its textual description based on the deep recurrent architecture by understanding the activity in the image ensuring that the machine has to generate syntactically and semantically correct sentences using deep learning models. The model is trained to maximize the likelihood of the target description sentence given the training.

**BLOCK DIAGRAM**

Our block diagram consists mainly of Input image, CNN flowed by RNN and final output with generated captions.

****

Let us know the explanation of each block in detail and how it performs.

**Input Image**

The input image should be in JPEG format.

**CNN**

CNN stands for “Convolutional Neural Network” and it is basically used for feature extraction and can produce a rich representation of input image by embedding it to a fixed length vector, such that this representation can be used for a variety of vision tasks. Hence it is natural to use a CNN as an “encoder” by first pre-training it for an image classification task and using the last hidden layer as an input t RNN (LSTM).

**LSTM**

LSTM stands for Long Short-term memory is an artificial recurrent neural network which is used to generate a sequence of words. LSTM is capable of keeping track of the objects that already have been described using text. The next words of LSTM are based on the current time step and previous hidden state. This process continues until it gets a end token of sentence. We also do batch normalization to improve the performance and accuracy of the model.

**GENERATED CAPTIONS**

Finally, the output is generated with textual description of the image.

**TECHNICAL STACK**

* Python
* CNN and LSTM model
* Required packages:

1.Numpy

2.Tensorflow

3. Bazel

4.NLTK

5.punkt

* PyCharm
* Google Colab
* Web framework:

1.Django

2. HTML/CSS

**IMPLEMENTATION**

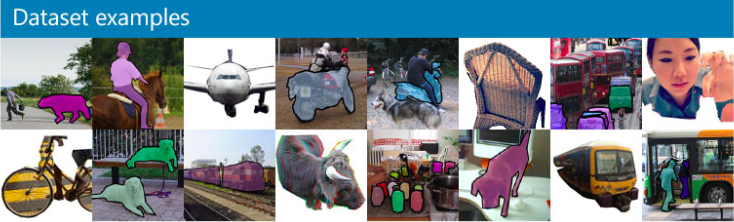
**Show and tell Model**

This model is a best example of an encoder – decoder neural network. It works by first encoding an image into a fixed length vector representation and then decoding the representation into a natural language description. This type of neural network is especially used for currently state-of-the-art for object recognition and detection for image encoder. For decoder LSTM is used for language modelling and machine translation.

**DATASET**

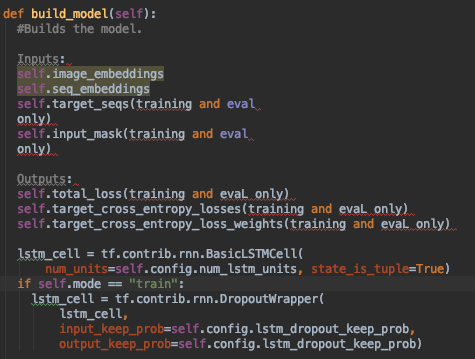
We have used MSCOCO dataset which is specifically used for image recognition, segmentation and captioning. It has more than 3,00,000 images where 80,000 images are used for training, 40,000 are used for validation and 30,000 images are used for testing. It has 80 object categories, more than 2 million instances and 5 captions per image.

Sample dataset:



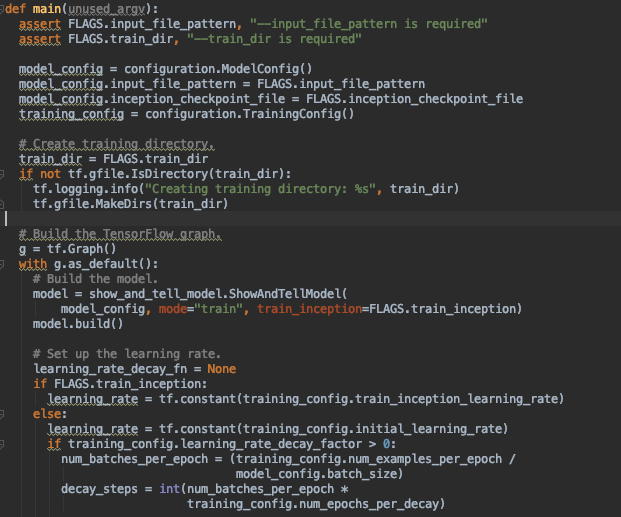
**BUILD THE MODEL**

Building the model using input sequence embeddings and image embeddings.

****

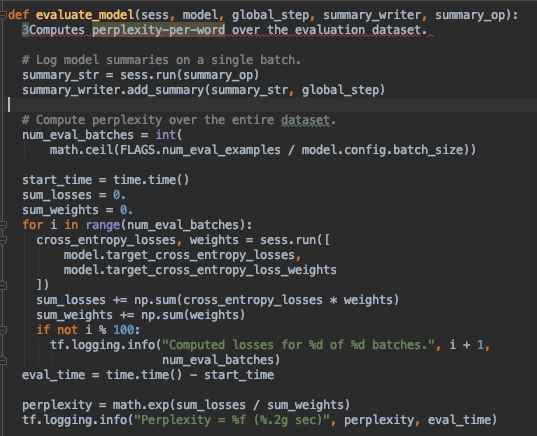
**TRAINING THE MODEL**

Train the model by creating the dictionary, setting up the learning rate, setting up the training ops, setting up the saver for saving and restoring model checkpoints.

****

**EVALUATING THE MODEL**

Evaluating the model by computing the perplexity per word over the entire dataset and log perplexity to the file writer.



**CAPTION GENERATOR**





**FRONT END**

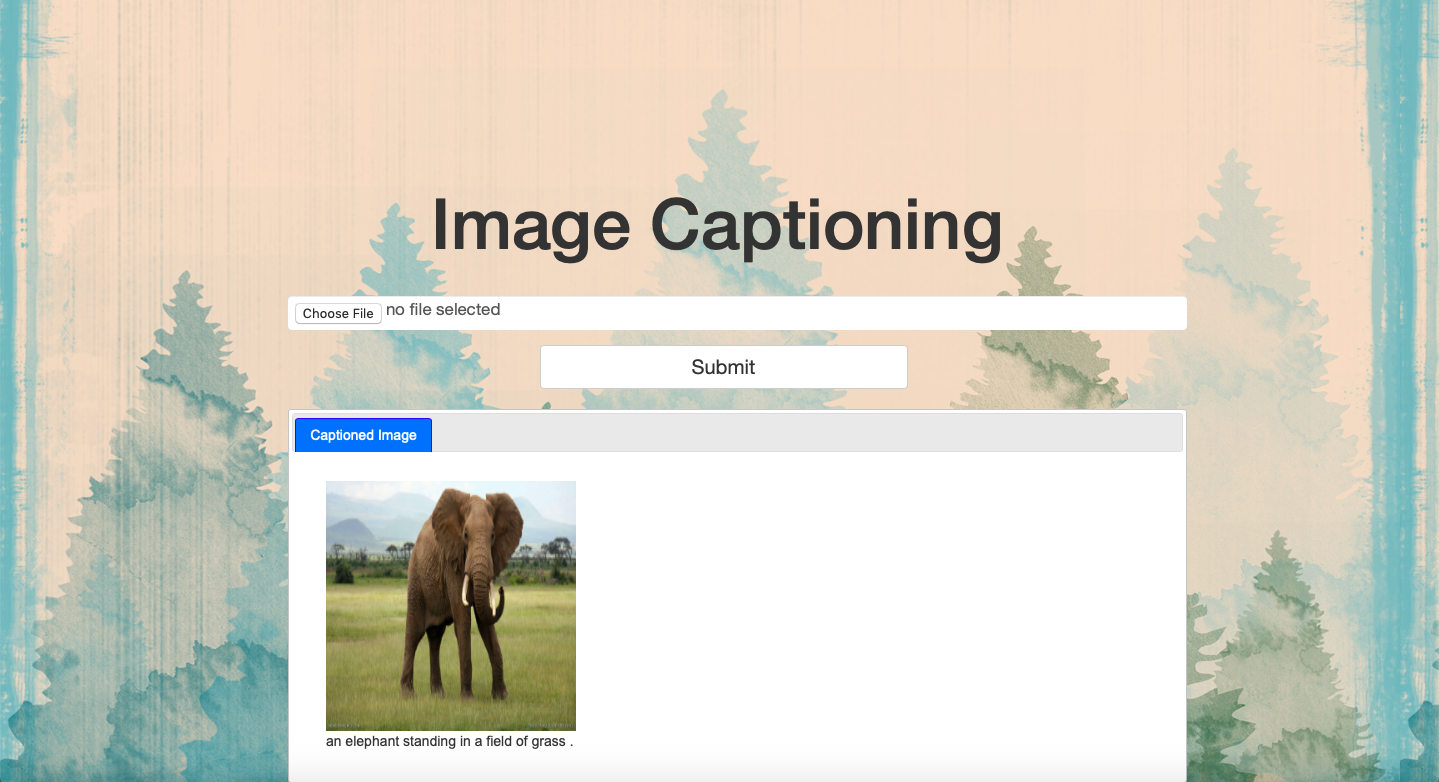
**Django**

It is python-based open source web framework which follows the model-template-view architectural pattern. It is used to ease the work of complex database driven websites.





After choosing the input image, click on the submit button and the output is displayed as shown below.



**SOURCE CODE**

Please go through the GitHub link for the source code.

**FUTURE WORK**

will keep the project going by enhancing the dense captioning, with better accuracy and performance.

**CONTRIBUTIONS**

**Naga Srividya Varanasi**- I did with the building and training of the model initially, then it followed with evaluating with appropriate metrics and generating captions. I also did with the front-end using Django, HTML/CSS.

**ChandraSekhar Pentakota**- helped with Django integrating with HTML/CSS.

**Pujitha-** helped with solving errors during execution and doing the documentation.

**REFERENCES**

* [**https://medium.freecodecamp.org/building-an-image-caption-generator-with-deep-learning-in-tensorflow-a142722e9b1f**](https://medium.freecodecamp.org/building-an-image-caption-generator-with-deep-learning-in-tensorflow-a142722e9b1f)
* [**https://arxiv.org/pdf/1609.06647.pdf**](https://arxiv.org/pdf/1609.06647.pdf)
* [**https://arxiv.org/pdf/1810.04020.pdf**](https://arxiv.org/pdf/1810.04020.pdf)