IMAGE CAPTION GENERATION FOR VISUALLY IMPAIRED

Varsha Khanna

UIN : 228000216

Abstract: The ability to recognize image features and generate accurate, syntactically reasonable text descriptions is important for many tasks in computer vision. Auto-captioning could, for example, be used to provide descriptions of website content, or to generate frame-by-frame descriptions of video for the vision-impaired. In this project, a multimodal architecture for generating image captions is explored. The architecture combines image feature information from a convolutional neural network with a recurrent neural network language model, in order to produce sentence-length descriptions of images. An attention mechanism is used, which allows the network to focus on the most relevant image features at each time-step during caption generation.

1. Introduction –

The paper" Show and Tell: A Neural Image Caption Generator" attempts to solve the problem of Automatically describing the content of an image. For problems like Machine Translation, the state-of-the-art solution was to use an RNN with an encoder and which encoded the input to a
fixed length vector representation and then have a decoder learn the mapping from that to whatever the required output was. This paper suggests an elegant solution to the problem
of image captioning making use of a modified version of the technique. This approach solves the issue of being able to describe previously unseen compositions of objects which were observed individually in the training data. The solution suggested here is to make use of a very deep convolutional net to generate the vector representation which is then used as an input to the RNN decoder, which, given the training data, learns the mapping from an image to a sequence of words, which is an 'end to end' network, which is trainable by stochastic gradient descent.

2. Related Work –

Human beings usually describe a scene using natural languages which are concise and compact.
However, machine vision systems describe the scene by taking an image which is a two-dimension arrays. From this perspective, Vinyal et al. (Vinyals et al.,) models the image captioning problem as a language translation problem in their Neural Image Caption (NIC) generator system. The idea is mapping the image and captions to the same space and learning a mapping from the image to the sentences. Donahue et al. (Donahue et al.,) proposed a more general Long-term Recurrent Convolutional Network (LRCN) method. The LRCN method not
only models the one-to-many (words) image captioning, but also models many-to-one action generation and many-to-many video description. They also provide publicly available

implementation based on Caffe framework (Jia et al., 2014), which further boosts the research on image captioning.

3. Dataset –

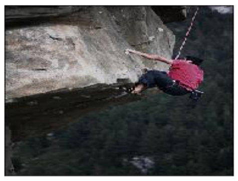
Flickr8k- Flickr8k consists of 8,108 hand-selected images from Flickr, deposit consists of 1000 images from 20 object classes, with each class containing 50 objects. The images were taken from 2008 PASCAL development toolkit and were annotated by Turkers who have cleared the qualification test designed by Rashtchian et al. (2010), with each image having five annotations, citing actions and events. Each image consists of 5 captions, annotated by Turkers who have cleared the qualification test designed by Rashtchianetal. (2010)

man on a bicycle riding on only one wheel .
asian man in orange hat is popping a wheelie on his bike .
a man on a bicycle is on only the back wheel .
a man is doing a wheelie on a mountain bike .
a man does a wheelie on his bicycle on the sidewalk .

five people are sitting together in the snow .
five children getting ready to sled .
a group of people sit in the snow overlooking a mountain scene .
a group of people sit atop a snowy mountain .
a group is sitting around a snowy crevasse .

a white crane stands tall as it looks out upon the ocean .
a water bird standing at the ocean 's edge .
a tall bird is standing on the sand beside the ocean .
a large bird stands in the water on the beach .
a grey bird stands majestically on a beach while waves roll in .

woman writing on a pad in room with gold , decorated walls .
the walls are covered in gold and patterns .
a woman standing near a decorated wall writes .
a woman behind a scrolled wall is writing
a person stands near golden walls .



a rock climber practices on a rock climbing wall .
a rock climber in a red shirt .
a person in a red shirt climbing up a rock face covered in assist handles .
a man is rock climbing high in the air .
a man in a pink shirt climbs a rock face

Fig. 1

4. Preprocessing –

We cleaned the captions and extracted image features provided in the dataset for further analysis.
PREPARING CAPTIONS :-

4.1 The caption dataset contains punctuations, singular words and numerical values that need to be cleaned before it is fed to the model because uncleaned dataset will not create good captions for the images. We then plotted the top 50 most common and least common word appearing in the resulting vocabulary.

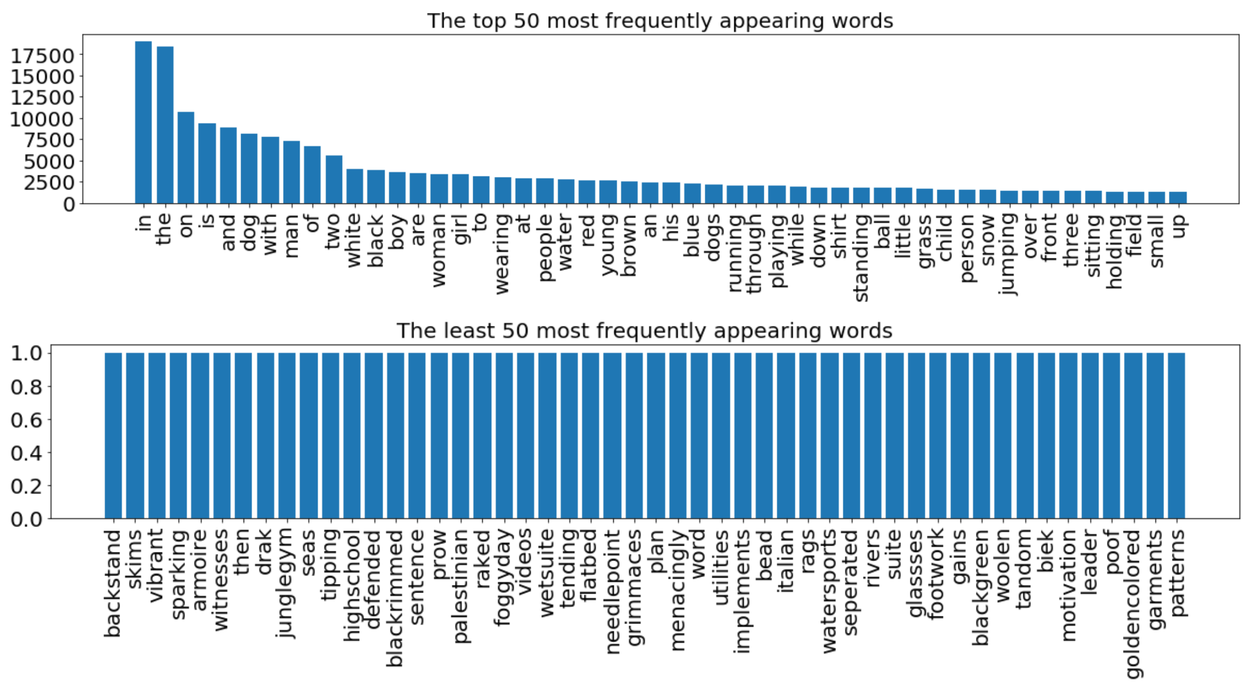


Fig. 2

4.2 Adding start and end sequence tokens for each caption

Start and end sequence must be added to the tokens so that it is easier to identify the captions for the images as each of them are of different length.

PREPARING IMAGES:-

4.3 Image Embedding—

The VGG16 is pre-trained on the ImageNet dataset. The last layer of the VGG-16 is excluded here because we are just using it for extracting the features rather than using for object classification. 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 that we will have done it once in advance.

This is an optimization that will make training our models faster and consume less memory.

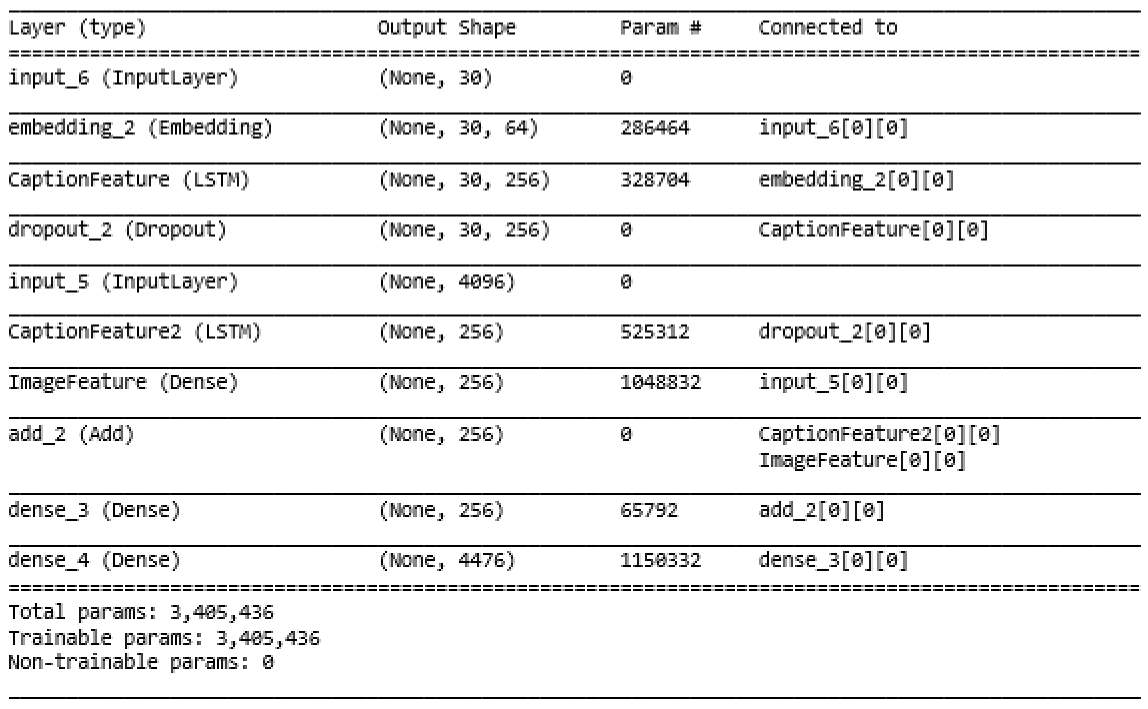
The image features are a 3-dimensional array with the shape (7, 7, 512).

5. Model Details –

The model involves three parts:

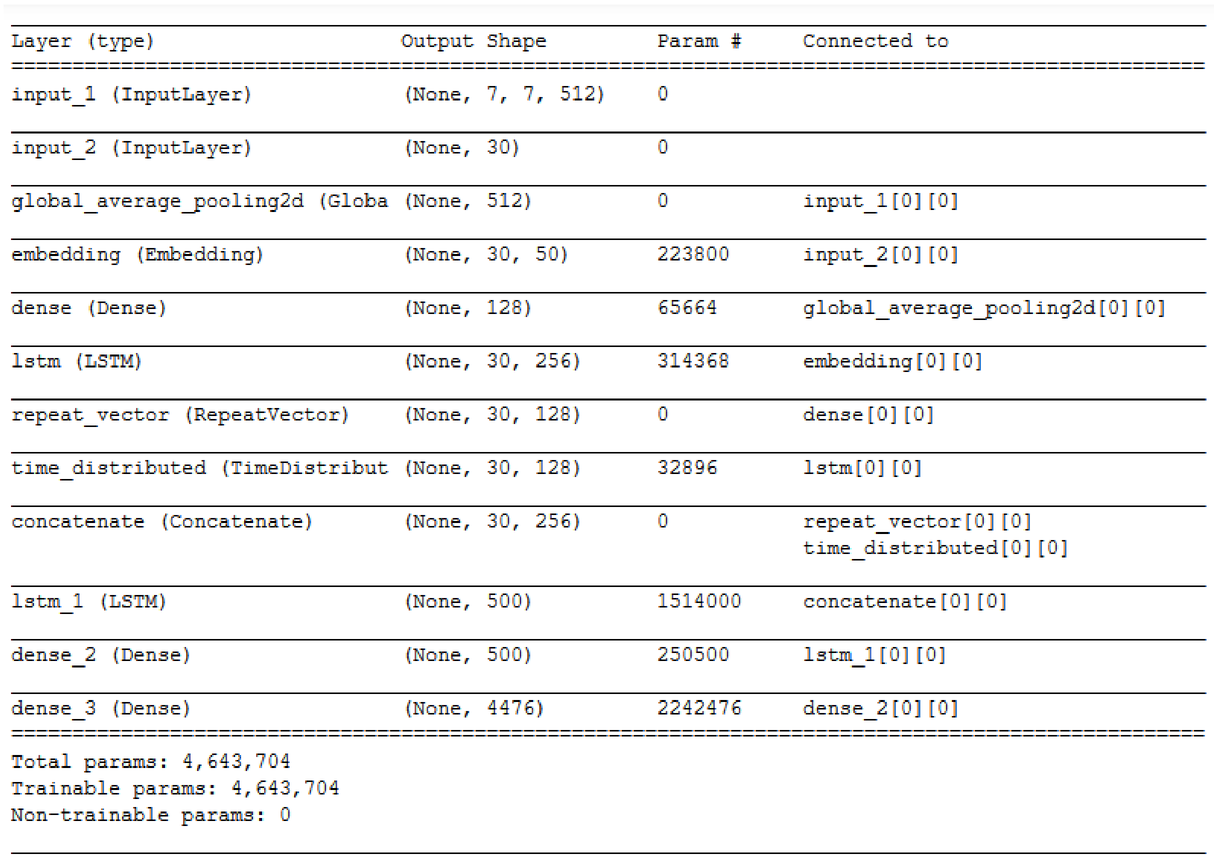
1. Photo Feature Extractor. This is a 16-layer VGG model pre-trained on the ImageNet dataset. We have pre-processed the photos with a the VGG model (without the top) and will use the extracted features predicted by this model as input.
2. Sequence Processor. This is a word embedding layer for handling the text input, followed by an LSTM layer. The LSTM output is interpreted by a Dense layer one output at a time.
3. Interpreter (for lack of a better name). Both the feature extractor and sequence processor output a fixed-length vector that is the length of a maximum sequence. These are concatenated together and processed by an LSTM and Dense layer before a final prediction is made. A conservative number of neurons is used in the base model. Specifically, a 128 Dense layer after the feature extractor, a 50-dimensionality word embedding followed by a 256-unit LSTM and 128 neurons Dense after the sequence processor, and finally a 500 unit LSTM followed by a 500 neuron Dense at the end of the network.

The model predicts a probability distribution across the vocabulary, therefore a softmax activation function is used and a categorical cross entropy loss function is minimized while fitting the network and optimization technique used is adam.



```
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
input_6 (InputLayer)             (None, 30)            0
____________________________________________________________________________________________________
embedding_2 (Embedding)          (None, 30, 64)        286464      input_6[0][0]
____________________________________________________________________________________________________
CaptionFeature (LSTM)            (None, 30, 256)       328704      embedding_2[0][0]
____________________________________________________________________________________________________
dropout_2 (Dropout)              (None, 30, 256)       0           CaptionFeature[0][0]
____________________________________________________________________________________________________
input_5 (InputLayer)             (None, 4096)          0
____________________________________________________________________________________________________
CaptionFeature2 (LSTM)           (None, 256)           525312      dropout_2[0][0]
____________________________________________________________________________________________________
ImageFeature (Dense)             (None, 256)           1048832     input_5[0][0]
____________________________________________________________________________________________________
add_2 (Add)                      (None, 256)           0           CaptionFeature2[0][0]
                                                                   ImageFeature[0][0]
____________________________________________________________________________________________________
dense_3 (Dense)                  (None, 256)           65792       add_2[0][0]
____________________________________________________________________________________________________
dense_4 (Dense)                  (None, 4476)          1150332     dense_3[0][0]
====================================================================================================
Total params: 3,405,436
Trainable params: 3,405,436
Non-trainable params: 0
____________________________________________________________________________________________________
```

Fig. 4 Baseline Model



```
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            (None, 7, 7, 512)    0
__________________________________________________________________________________________________
input_2 (InputLayer)            (None, 30)           0
__________________________________________________________________________________________________
global_average_pooling2d (Globa (None, 512)          0           input_1[0][0]
__________________________________________________________________________________________________
embedding (Embedding)           (None, 30, 50)       223800      input_2[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 128)          65664       global_average_pooling2d[0][0]
__________________________________________________________________________________________________
lstm (LSTM)                     (None, 30, 256)      314368      embedding[0][0]
__________________________________________________________________________________________________
repeat_vector (RepeatVector)    (None, 30, 128)      0           dense[0][0]
__________________________________________________________________________________________________
time_distributed (TimeDistribut (None, 30, 128)      32896       lstm[0][0]
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 30, 256)      0           repeat_vector[0][0]
                                                                 time_distributed[0][0]
__________________________________________________________________________________________________
lstm_1 (LSTM)                   (None, 500)          1514000     concatenate[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 500)          250500      lstm_1[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 4476)         2242476     dense_2[0][0]
==================================================================================================
Total params: 4,643,704
Trainable params: 4,643,704
Non-trainable params: 0
__________________________________________________________________________________________________
```

Fig. 5 Improved Model

Difference from baseline:

- The image features are a 3-dimensional array with the shape (7, 7, 512) instead of flattened array of shape (4096,)
- GlobalAveragePooling2D to achieve average pooling. Global average pooling was developed to reduce overfitting for image classification problems but may offer some benefit in interpreting the features extracted from the image.
- Size of fixed Vector: the photo feature extractor and the text sequence encoder both output a 128 element vector. These vectors are then concatenated to be processed by the language model. The 128-element vector from each sub-model contains everything known about the input sequence and photo. [Time Distributed Layer]
- Embedding size for language model changed from 64 to 50.

Hyperparameters:

- No of Epochs : 15
- Batch size : 64
- Train-Test Split Criteria : 80-20%
- Loss: categorical_cross_entropy

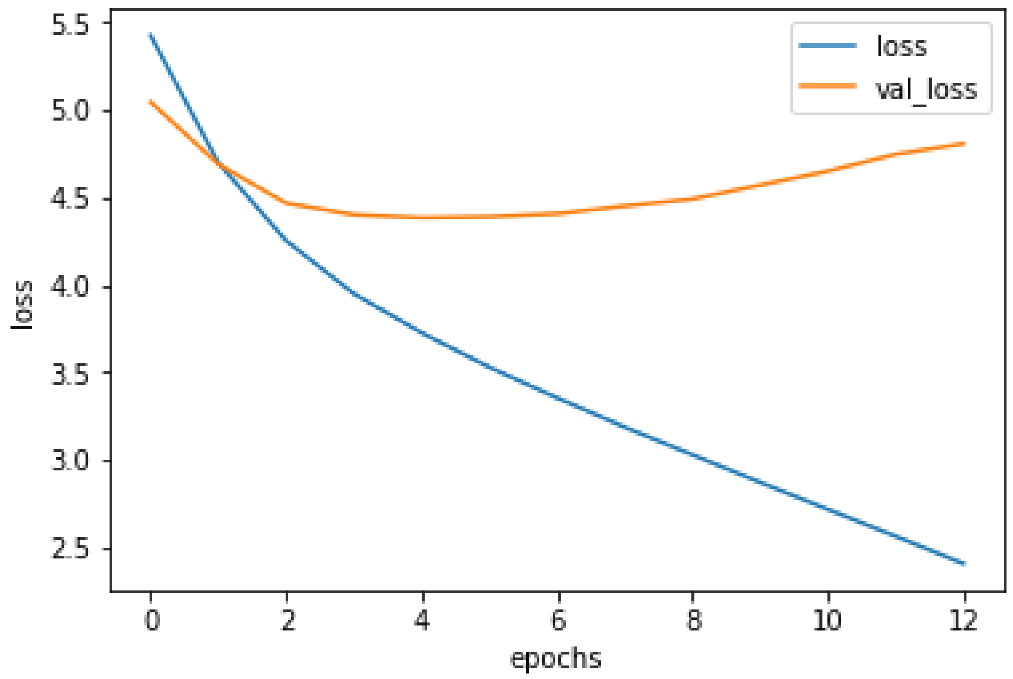
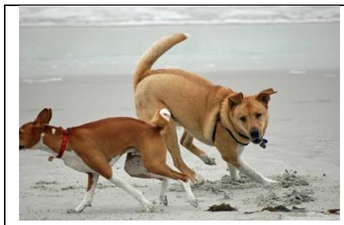
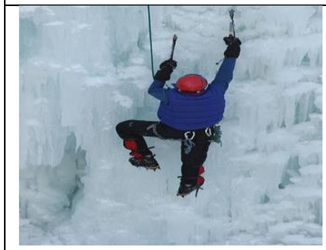


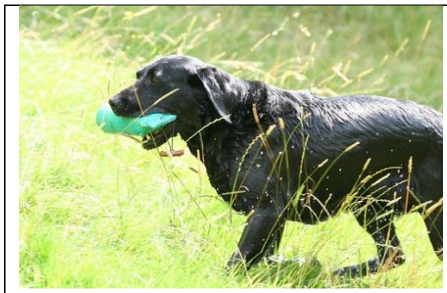
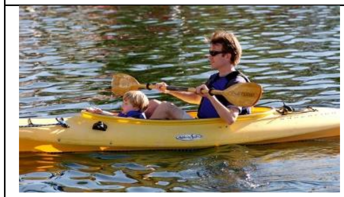
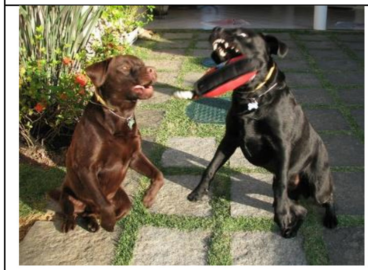
Fig. 6 Plotting validation and training loss over epochs

6. Evaluation

Evaluation metrics We use BLEU (Papineni et al., 2002) to measure the similarity of the captions generated by our method and human beings. BLEU stands for Bilingual Evaluation Understudy; it is a score for comparing a candidate text to one or more reference text. The unigram scores (B-1) account for the adequacy of the translation, while longer n-gram scores (B-2, B-3, B-4) account for the fluency.

Some Images evaluated by our model are : -

	`startseq two dogs are standing on the beach endseq`
	`startseq man in flannel snowboarding endseq`

	startseq black dog digging through the woods endseq
	startseq child and woman are jumping into the pool endseq
	startseq black dog is playing with blue hind ball endseq

MODEL	BLEU
Baseline	0.11
Improved	0.35

7. Conclusion

The model has been successfully trained to generate the captions as expected for the images. The caption generation has constantly been improved by fine tuning the model with different hyper parameter. Higher BLEU score indicates that the generated captions are very similar to those of the actual caption present on the images. With the help of Tensorboard, we were able to see how different training process had an impact on the model. The validation loss falls up to 6th epoch and then increases afterwards, while the training loss still continues falling

The following were the major outcomes and observations of the training process and testing the model on the test data:

- The validation loss increases after 5\6th epoch in most cases even though the training loss decreases over time. This indicates that the model is over fitting and the training needs to stop.
- Higher BLEU score doesn't always translate to better generated captions. If the model overfits on your training data, it will lead the model to go through details in the image and generate out captions which don't make sense. It can be seen in the good and the bad captions generated above.
- GlobalAverageMaxPooling takes the major features of an image and passes it for convolution.

8. Future Scope

We can use word embeddings using word to vec to tokenize our captions text so as to give probabilities to words rather than dictionary indexes. Word2Vec also implies semantic meaning to words.
This can improve our BLEU score.

9. Instructions to run the demo

Dependencies

- Keras
- Tensorflow GPU
- Pre-trained VGG-16 weights
- NLTK
- Matplotlib

Execution

Run gui gui.py and play around with the UI

Code

Github link : https://github.com/vkhanna92/Image-Caption-Generator-DNNs

Video Link

YouTube Link : https://youtu.be/gjSIUrqGits STEPS:

1. Download .rar file and unzip it in a directory.

2. Open command line in the same directory and type following command: python gui.py
3. Click on Upload Image button and select any size image to view properly in the demo.
4. Click on Predict button to view results