BMS College of Engineering
Department of Information Science and Engineering
SEE Review

Deep Learning Image Caption Generator

P Aishwarya Naidu (1BM16IS062)
Gehna Anand (1BM16IS034)
Satvik Vats (1BM16IS079)

Project Guide:-
Nalina V
Assistant Professor, Department of ISE,BMSCE

Problem Identification

- Digital image processing has become important and economical in many fields like signature recognition, iris recognition and face recognition, in forensics, in automobile detection systems and in military applications across the world.

- Each of these applications has its special basic requirements, which may be unique from the others.

- Any stakeholder of such systems or models is concerned and demands their system to be faster, more accurate than other counterparts as well as cheaper and equipped with more extensive computational powers.

- The problem introduces a captioning task, which requires a computer vision system to both localize and describe salient regions in images in natural language.

- A computer vision based model that is unbiased and free of any prejudice towards anything or anyone is required to generate a caption describing the images given to it as input.

- So that such description can be used to automate existing systems like traffic control systems, flood control systems or surveillance systems.

- This will reduce chances of errors in such critical works and also the surveillance can be conducted 24X7 without human interaction.

Applications

- Self-driving cars — Automatic driving is one of the biggest challenges and if we can properly caption the scene around the car, it can give a boost to the self-driving system.

- Aid to the blind — We can create a product for the blind which will guide them travelling on the roads without the support of anyone else. We can do this by first converting the scene into text and then the text to voice. Both are now famous applications of Deep Learning.

- CCTV cameras are everywhere today, but along with viewing the world, if we can also generate relevant captions, then we can raise alarms as soon as there is some malicious activity going on somewhere. This could probably help reduce some crime and/or accidents.

- Automatic Captioning can help make Google Image Search as good as Google Search, as then every image could be first converted into a caption and then search can be performed based on the caption.

A person on a beach flying a kite.

A black and white photo of a train on a train track.

A person skiing down a snow covered slope.

A group of giraffe standing next to each other.

Fig. 1- Image- Caption Examples

Project Objectives

- Using Long-Short Term Memory (LSTM) to generate sentences in natural languages that will combine words from vocabulary based on the different focus areas of the input image and make sure that the words in the sentences are all related and makes perfect sense.
- Demonstrate successful use of Recurrent Neural Networks (RNNs) to generate captions for input images in the system in natural language (English).
- To get high accuracy in correctly describing an input image.

LITERATURE SURVEY

Show, Attend and Tell: Neural Image Caption Generation with Visual Attention

- Kelvin Xu et al. have introduced an attention based model that automatically learns to describe the content of images.

- The authors describe an attention based model. Rather than compress an entire image into a static representation, attention allows for salient features to dynamically come to the forefront as needed. This is especially important when there is a lot of clutter in an image.

- They attempt to incorporate a form of attention with two variants:
    - a “hard” attention mechanism
    - a “soft” attention mechanism

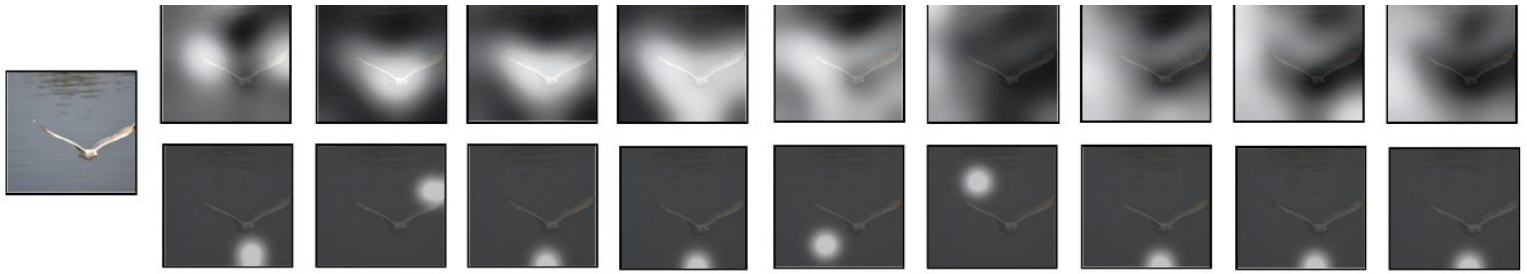


Figure 3. Examples of attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word)


A woman is throwing a frisbee in a park.


A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.


A little girl sitting on a bed with a teddy bear.

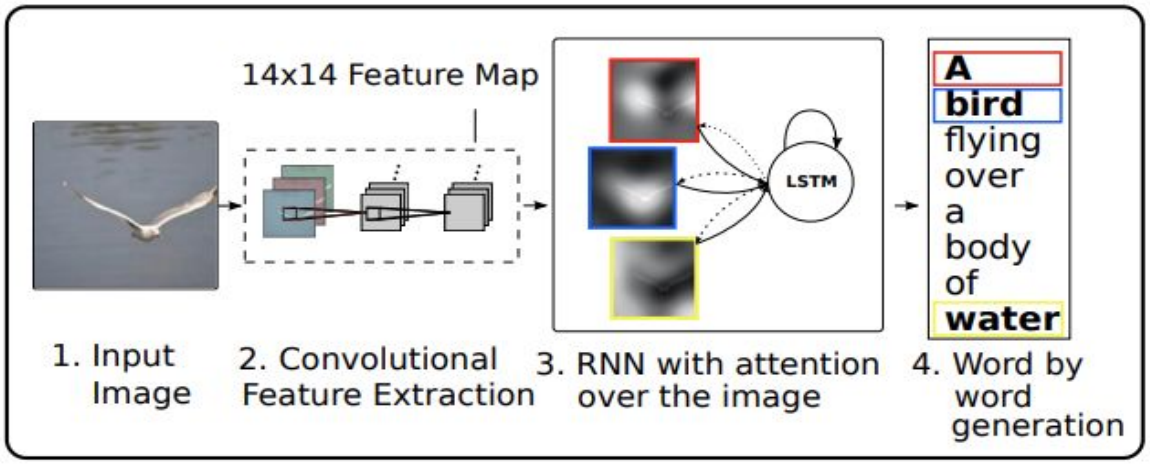

A group of people sitting on a boat in the water.


A giraffe standing in a forest with trees in the background.

Fig. 2- Attention Model [1]

- They show how we can gain insight and interpret the results of this framework by visualizing “where” and “what” the attention is focused on.

- Finally, they quantitatively validate the usefulness of attention in caption generation with state of the art performance on three benchmark datasets: Flickr 8k, Flickr 30k and the MS COCO dataset.



Composing Simple Image Descriptions using Web-scale N-grams

- Siming Li et. al. have introduced an image description model that tries to make more human-like annotations than past methodologies.

- This methodology comprises two stages: (n-gram) phrase selection and (n-gram) phrase fusion.

- The initial step "phrase selection" involves gathering candidate phrases that might be conceivably helpful for producing the description of a given image.

- The second step – phrase fusion – finds the ideal compatible set of phrases utilizing dynamic programming to make another (and increasingly unpredictable) state that depicts the image.

- For a given image,the image recognizer extracts objects, attributes and spatial relationships among objects.

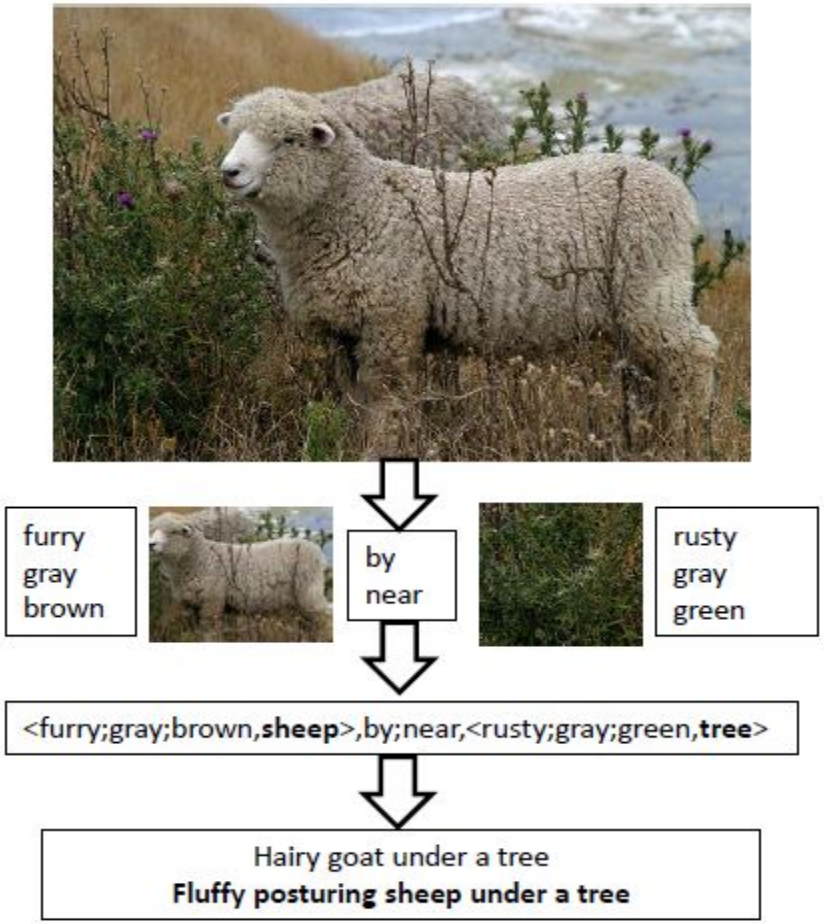


Fig. 4- N-grams to Image Descriptions [2]

System Design

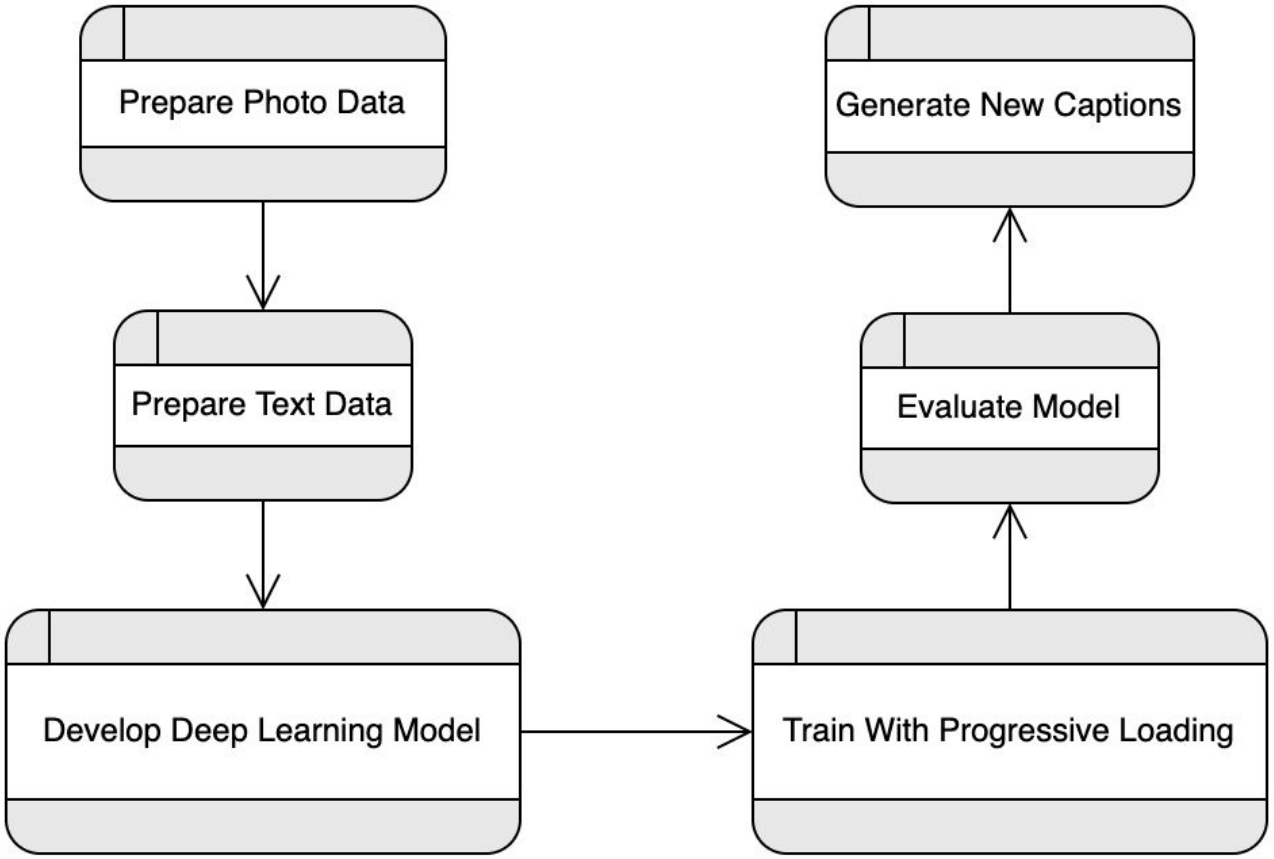


Fig. 5- System Design

Low Level Design

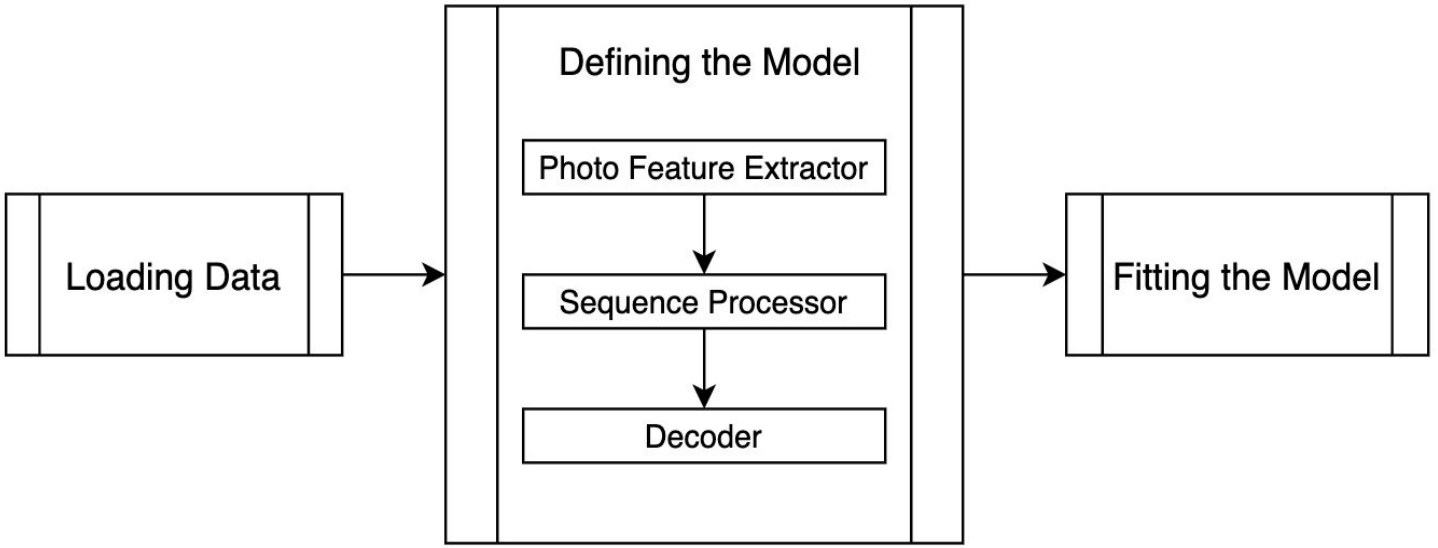


Fig. 6- The Development of Deep Learning Model (Low Level Design)

General Implementation Factors

- **Tools Used**
  - **Python**
  - **Anaconda**
  - **Keras**

Specific Implementation Details

Step 1: Data Collection

Dataset used - MSCOCO

The dataset contains 82000 images and each image is associated with at least 5 captions. To quicken training we randomly selected 30000 datapoint. The images are split as follows: Training Set (24000 images) and Test Set (6000 images)

man laying on bench holding leash of dog sitting on ground
a shirtless man lies on a park bench with his dog .
a man sleeping on a bench outside with a white and black dog sitting next to him .
a man lays on the bench to which a white dog is also tied .
a man lays on a bench while his dog sits by him .

the man with pierced ears is wearing glasses and an orange hat .
a man with glasses is wearing a beer can crocheted hat .
a man with gauges and glasses is wearing a blitz hat .
a man wears an orange hat and glasses .
a man in an orange hat starring at something .

Fig. 7- Data sample.

Step 2: Prepare photo data

- We use the InceptionV3 model which Keras provides directly.
- The “photo features” are pre-computed using the pre-trained model and saved to a file.
- These features can later be loaded and fed into the model as the interpretation of a given photo in the dataset.
- This optimization will make training the models faster and consume less memory.
- The InceptionV3 model is loaded in Keras using InceptionV3 class. The last layer from the loaded model is removed, as this is the model used to predict a classification for a photo. Only the internal representation of the photo is of importance. These are the “features” that the model has extracted from the photo.

Available models

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159

Step 3: Prepare text data

- The text is cleaned in the following ways in order to reduce the size of the vocabulary of words the model needs to work with:
  a. Convert all words to lowercase.
  b. Remove all punctuation.
  c. Remove all words that are one character or less in length (e.g. ‘a’).
  d. Remove all words with numbers in them.
- This is a word embedding layer for handling the text input, followed by a Long Short-Term Memory (LSTM) recurrent neural network layer which completes the sequence processing.
- Text Processing is done where the image descriptors of the image are converted to numerical representation.
- This embedding layer can refer to words from the captions in two different manners, that is, either frequency based or prediction based embedding.

Step 4: Develop deep learning model

Loading data

- The prepared photo and text data must be loaded so that it can be used to fit the model.
- The model generates a caption given a photo, and the caption will be generated one word at a time. The sequence of previously generated words will be provided as input.
- Therefore, there is a need for a ‘first word’ to begin the generation process and a ‘last words’ to indicate the end of the caption. The string ‘startseq’ and ‘endseq’ are used for this. These tokens are added to the loaded descriptions as they are loaded.

- The description text will need to be encoded to numbers before it can be presented to the model as an input or compared to the model's predictions.
- The first step in encoding the data is to create a consistent mapping from words to unique integers values using the Tokenizer class.

This is how the model will be trained.

```
X1,     X2 (text sequence),                     y (word)
photo   startseq,                               little
photo   startseq, little,                       boy
photo   startseq, little, boy,                  walking
photo   startseq, little, boy, walking,         on
photo   startseq, little, boy, walking, on,     road
photo   startseq, little, boy, walking, on, road,   endseq
```

Fig. 8- Training the Model

When the model is used to generate descriptions, the generated words will be concatenated and recursively provided as input to generate a caption for an image.

There are two input arrays to the mode: one for photo features and one for the encoded text.

There is one output for the model which is encoded next word in the text sequence.

The photo features will be fed directly to another part of the model. The model will output a prediction, which will be a probability distribution over all the words in the vocabulary.

Defining the model

A standard encoder-decoder recurrent neural network architecture is used to address the image generation problem. This involved two elements:

1. Encoder: A network model that reads the photograph input and encodes the content into a fixed length vector using an internal representation.
2. Decoder: A network model that reads the encoded photograph and generates the textual description out.

The merge model combines both the encoded form of the image input with the encoded form of the text description generated so far. The combination of these two encoded inputs is then used by a very simple decoder model to generate the next word in the sequence.

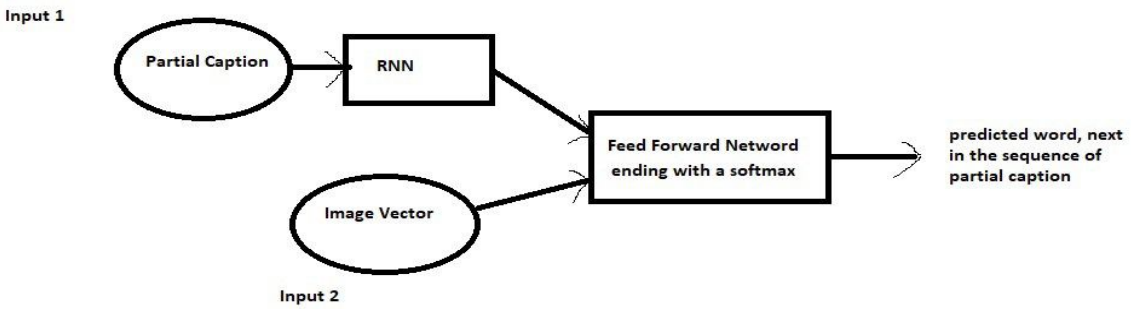


Fig. 9- Defining the model

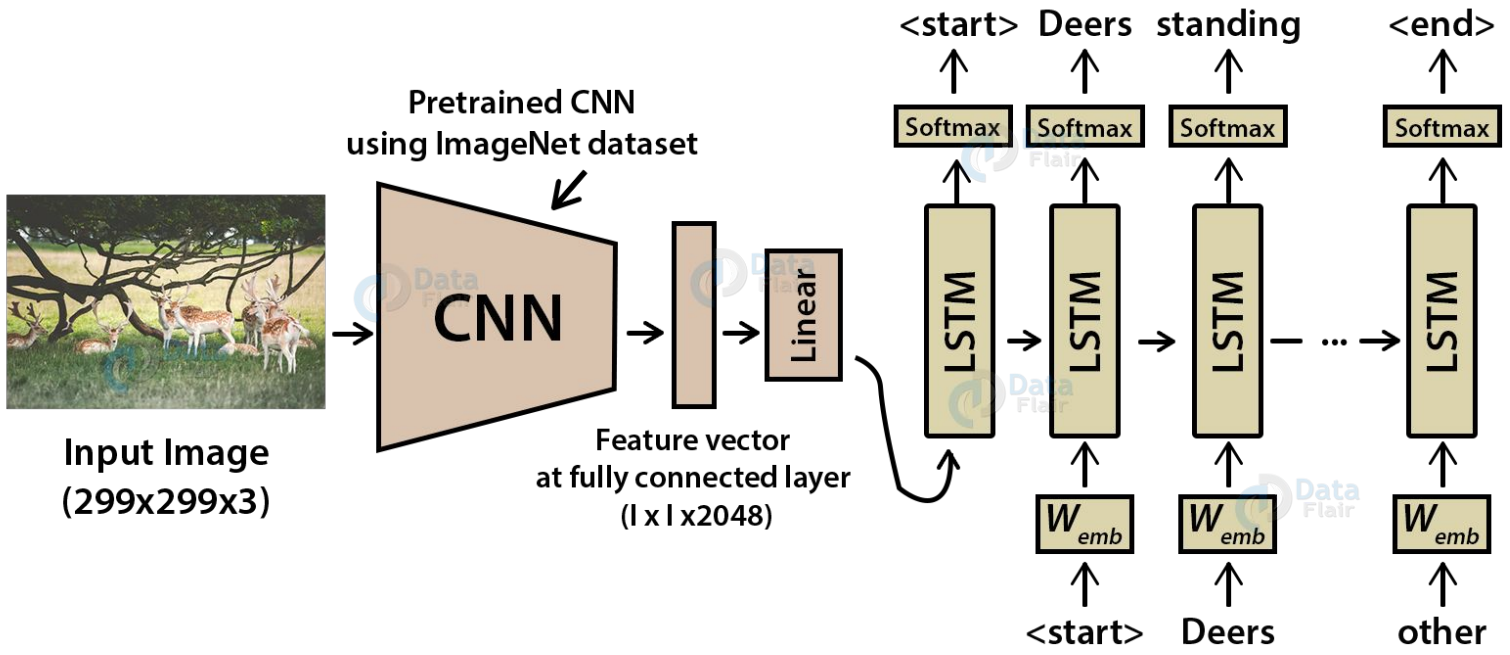
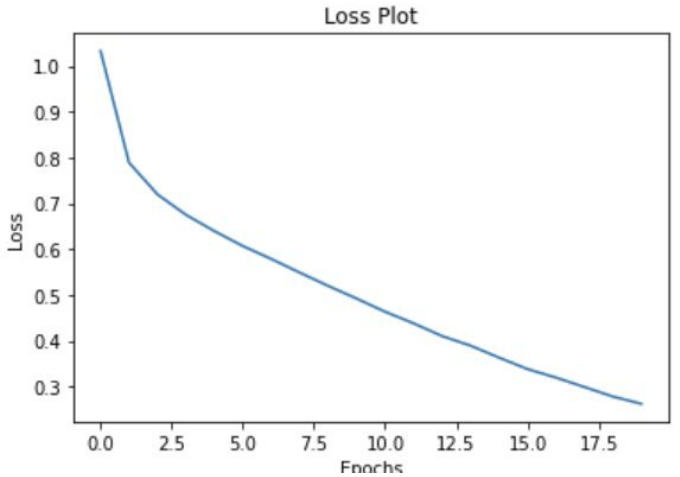


Fig. 10- Model design

Fitting the model

- The model learns fast and quickly overfits the training dataset. For this reason, the trained model should be monitored on the holdout dataset. When the skill of the model on the development dataset improves at the end of an epoch, the whole model will be saved to a file.

- At the end of the run, the saved model with the best skill on the training dataset can be used as the final model. This is done by defining a *ModelCheckpoint* in Keras and specifying it to monitor the minimum loss on the dataset.




```
optimizer = tf.keras.optimizers.Adam()
loss_object = tf.keras.losses.SparseCategoricalCrossentropy(
    from_logits=True, reduction='none')

def loss_function(real, pred):
  mask = tf.math.logical_not(tf.math.equal(real, 0))
  loss_ = loss_object(real, pred)

  mask = tf.cast(mask, dtype=loss_.dtype)
  loss_ *= mask

  return tf.reduce_mean(loss_)
```

Fig. 12 -Loss Plot

Step 5: Train with progressive loading

- Keras supports progressively loading datasets by using a generator function on the model. A generator is the term used to describe a function used to return batches of samples for the model to train on.
- A data generator will yield one photo's worth of data per batch. This will be all of the sequences generated for a photo and its set of descriptions.
- The big memory saving it offers is to not have the unrolled sequences of train and test data in memory prior to fitting the model, but that these samples are created as needed per photo.

Evaluating the Model

- We have evaluated the model by generating descriptions for all photos in the test dataset and evaluating those predictions with a standard cost function.

- 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.

Evaluating the model

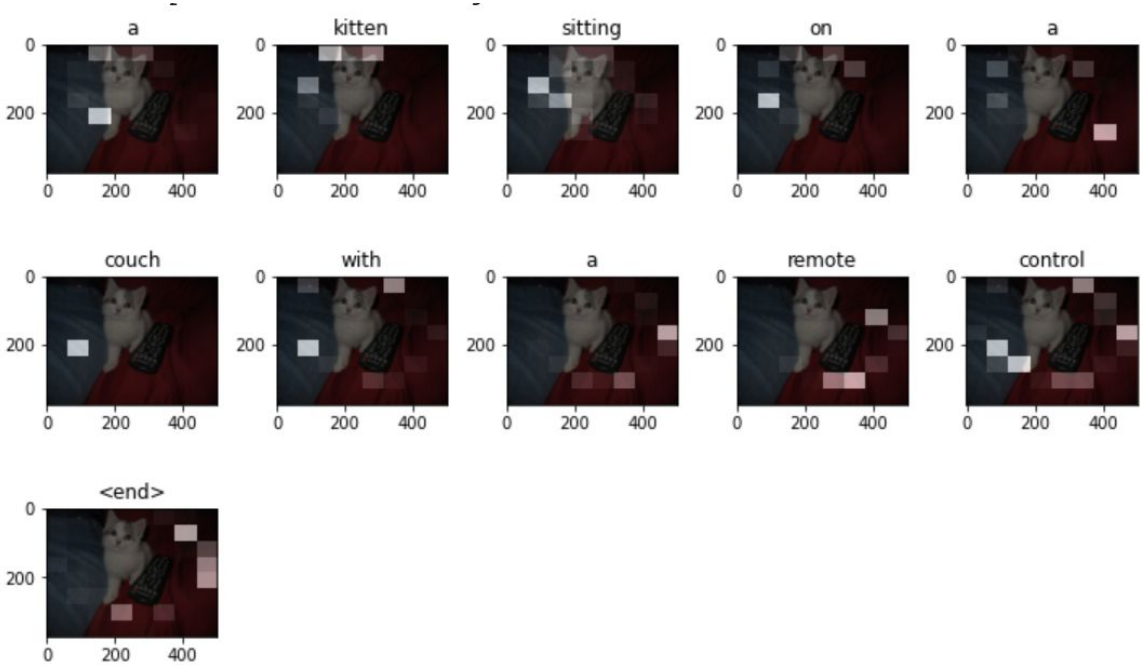
- The trained model is evaluated against a given dataset of photo descriptions and photo features.

- The actual and predicted descriptions are collected and evaluated collectively using the corpus BLEU (bilingual evaluation understudy) 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.

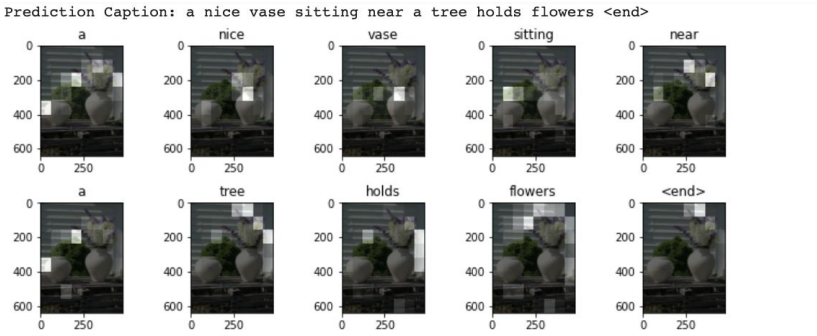
Objective :1

- **Using Long-Short Term Memory (LSTM) to generate sentences in natural languages that will combine words from vocabulary based on the different focus areas of the input image and make sure that the words in the sentences are all related and makes perfect sense.**



Objective :2

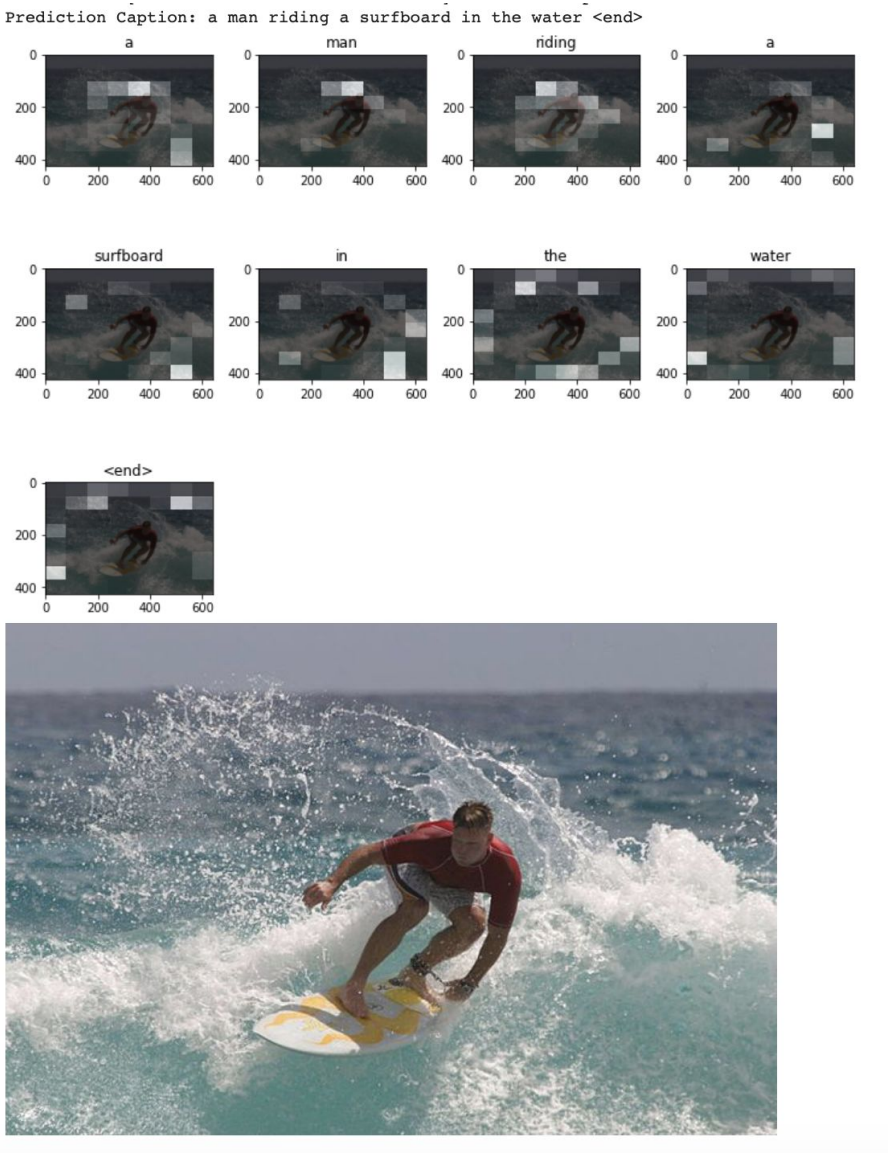
- **Demonstrate successful use of Recurrent Neural Networks (RNNs) to generate captions for input images in the system in natural language (English).**



A

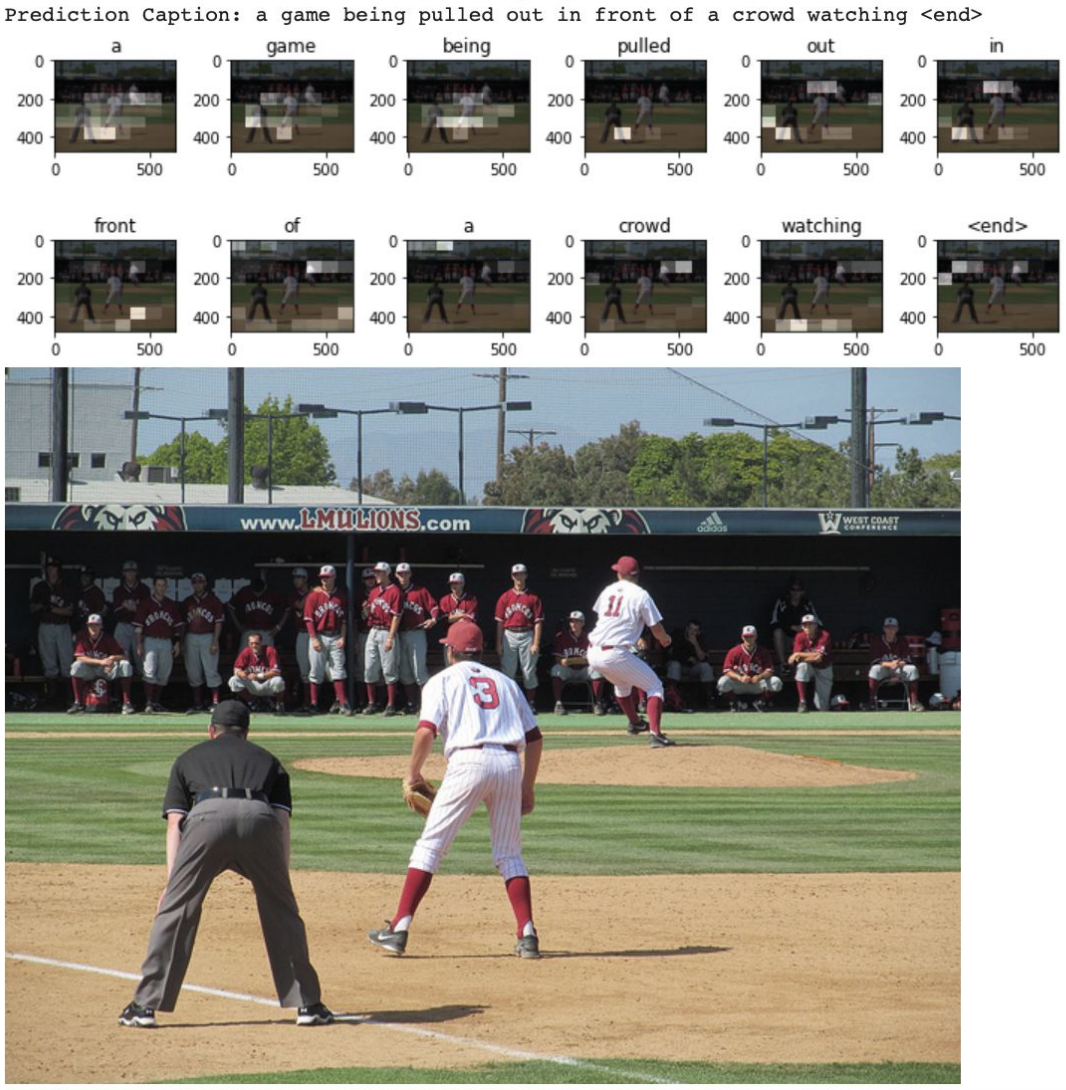
nice vase sitting near a tree holds flower

Fig. 13- Images for demonstration



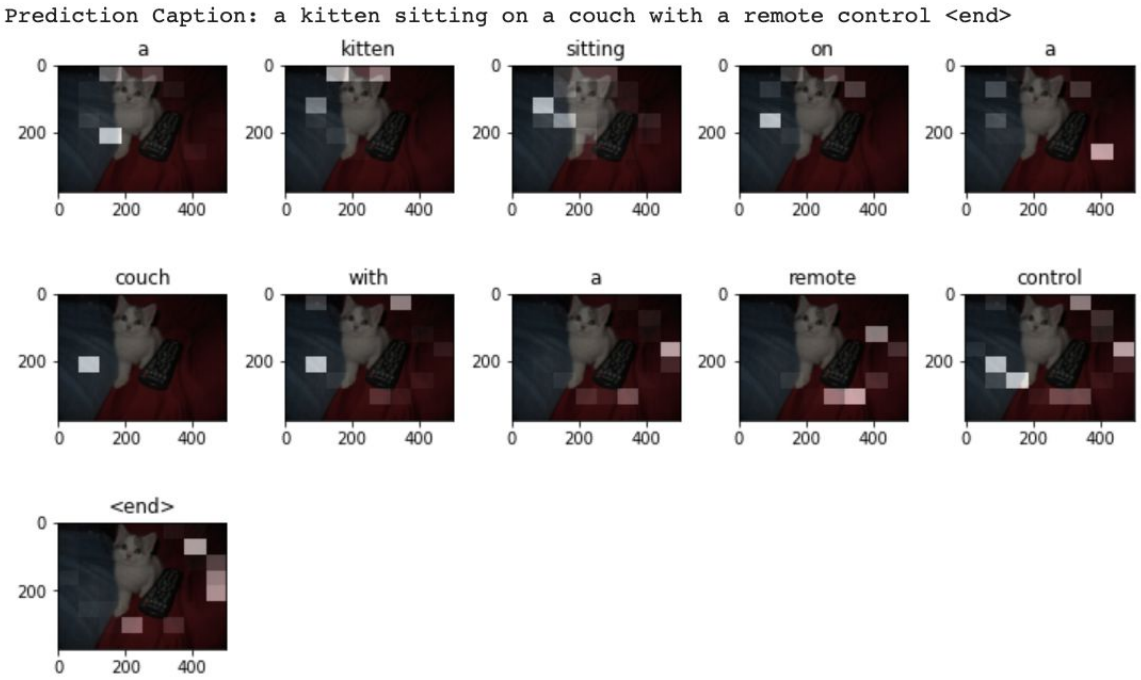
A man riding a surfboard in the water

Fig. 14- Image for demonstration



A game being pulled out in front of a crowd watching

Fig. 16- Image for demonstration



A kitten sitting on a couch with a remote control

Fig. 18- Image for demonstration

Objective :3

- **To get high accuracy in correctly describing an input image. (Testing the developed model)**

```
from nltk.translate.bleu_score import corpus_bleu
def evaluate_model():
  actual, predicted = list(), list()

  len_val = len(img_name_val)
  for rid in range(0, len_val):
    image = img_name_val[rid]
    real_caption = ' '.join([tokenizer.index_word[i] for i in cap_val[rid] if i not in [0]])
    result, attention_plot = evaluate(image)

    actual.append([real_caption.split()])
    predicted.append(result)



  print('BLEU-1: %f' % corpus_bleu(actual, predicted, weights=(1.0, 0, 0, 0)))
  print('BLEU-2: %f' % corpus_bleu(actual, predicted, weights=(0.5, 0.5, 0, 0)))
  print('BLEU-3: %f' % corpus_bleu(actual, predicted, weights=(0.3, 0.3, 0.3, 0)))
  print('BLEU-4: %f' % corpus_bleu(actual, predicted, weights=(0.25, 0.25, 0.25, 0.25)))

evaluate_model()
```

```
BLEU-1: 0.372867
BLEU-2: 0.186140
BLEU-3: 0.113631
BLEU-4: 0.045331
```

Fig. 19 - Code snippet & Results

Thank you!