ECE 285 – MLIP – Project A
Image Captioning

Written by Raghav Kalayanasundaram Subramanian. Last Updated on October 22, 2019.

The goal of this project is to automatically describe the content of an image. To accomplish this, we would need to identity one/multiple objects in the image, the way these relate to each other and any attributes or activities they are involved in. This extracted semantic knowledge is stored in a fixed length vector representation, known as an embedding. The embedding is used to generate a sentence in a known target language T. With recent advancements in Deep learning based models for Computer Vision and Natural Language Processing tasks, it has become easier to connect vision and language to build models that help with scene understanding. One example of image captioning is the Microsoft caption bot .

Note that additional information may be posted on Piazza by the Instructor or the TAs. This document is also subject to be updated. Most recent instructions will always prevail over the older ones. So look at the posting/updating dates and make sure to stay updated.

1 Image Captioning

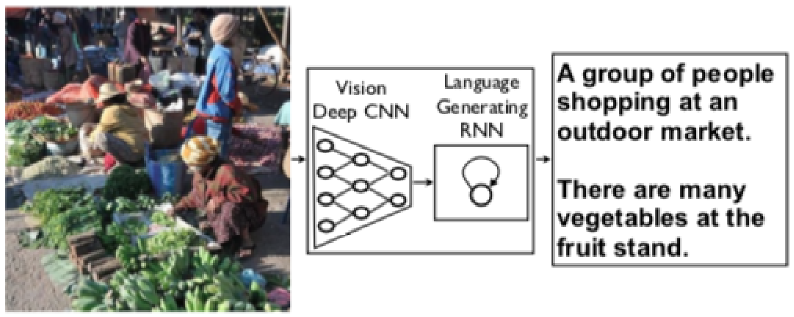

Vision
Deep CNN
Language
Generating
RNN
A group of people shopping at an outdoor market.
There are many vegetables at the fruit stand.

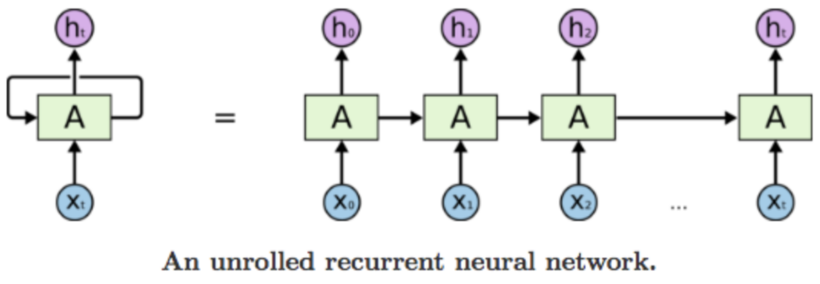
Figure 1: Image Captioning models with a Encoder-Decoder Framework

Image Captioning is the process of generating a textual description from an Image. With deep learning approaches to image captioning, semantic understanding across image data has increased. Most of the recent success in Image Captioning is derived from Deep learning models that adopt an encoder-decoder framework. The encoder-decoder framework was derived from the domain of machine translation, where the idea is to convert speech/text from a given language S to a target language T. For Image captioning, a convolutional neural network(CNN) is used as the encoder, to obtain a rich vectorial representation of the image with region-based visual features. This representation is fed to the decoder, a recurrent neural network (RNN) based caption decoder that iteratively generates output caption in natural language sentences.

2 Background

2.1 Recurrent neural networks (RNNs)

A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor. The core reason that recurrent nets are more exciting is that they allow us to operate over sequences of vectors: Sequences in the input, the output, or in the most general case both.



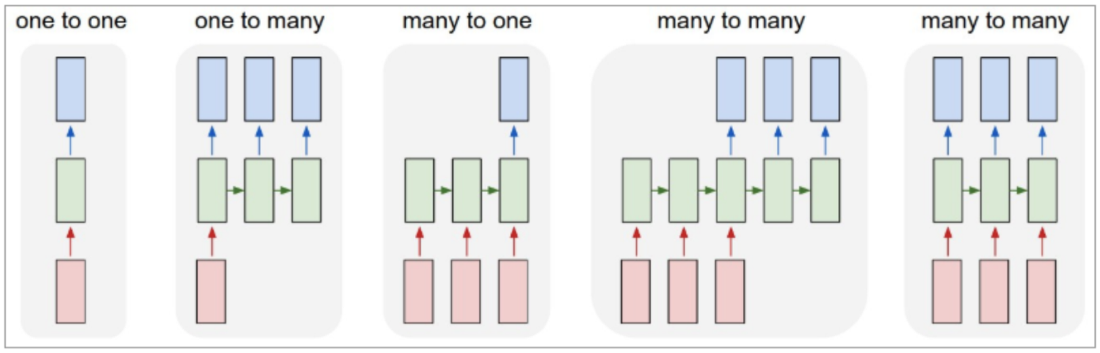


Figure 2: Recurrent Neural Networks

Each rectangle is a tensor and arrows represent functions (*e.g.*, matrix multiplications). Input vectors are in red, output vectors are in blue and green vectors hold the RNN's state (more on this soon). From left to right:

1. Vanilla mode of processing without RNN, from fixed-sized input to fixed-sized output (*e.g.*, image classification).

2. Sequence output (*e.g.*, image captioning takes an image and outputs a sentence of words).

3. Sequence input (*e.g.*, sentiment analysis where a given sentence is classified as expressing positive or negative sentiment).

4. Sequence input and sequence output (*e.g.*, Machine Translation: an RNN reads a sentence in English and then outputs a sentence in French).

5. Synced sequence input and output (*e.g.*, video classification where we wish to label each frame of the video).

2.2 Long Short Term Memory(LSTMs)

Long Short Term Memory networks (LSTMs) are special RNNs, capable of learning long-term dependencies. RNNs struggle with remembering information for a very long time and have the problem of vanishing and exploding gradients, which results in complexity during training. LSTMs use structures called gates to regulate the flow of information to memory cells, which encode the inputs observed at every time step till the current step. Gates are usually composed of a sigmoid layer with an output between 0 to 1, with 0 representing "no information through" and 1 representing "let all information through". There are three gates: Input (i), Output (o) and Forget (f) gates used to control if new input can be read, new cell value can be given as output, and whether the cell state can be forgotten/has to be retained. Let σ represent the sigmoid function and h represent the tanh function, and let $\odot$ represent the element-wise product between two matrices. Also, assume that W_{ij} represents trained parameters as part of the LSTM.

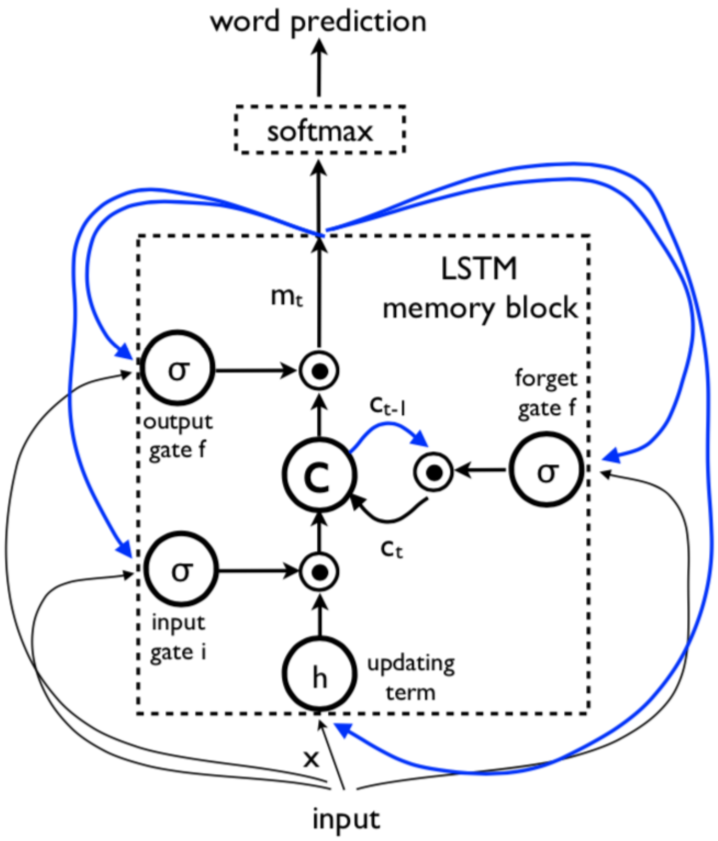


Figure 3: LSTMs

$$i_t = \sigma(W_{ix}x_t + W_{im}m_{t-1}) \tag{1}$$

$$f_t = \sigma(W_{fx}x_t + W_{fm}m_{t-1}) \tag{2}$$

$$o_t = \sigma(W_{ox}x_t + W_{om}m_{t-1}) \tag{3}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot h(W_{cx}x_t + W_{cm}m_{t-1}) \tag{4}$$

$$m_t = o_t \odot c_t \tag{5}$$

There are several other variants to LSTMs such as the Gated Recurrent Unit (GRU), Depth Gated RNNs, Clockwork RNNs etc. but these overall they help learn long term dependencies using different approaches.

Please visit `http://colah.github.io/posts/2015-08-Understanding-LSTMs/` for more understanding of RNN and LSTM Networks or refer to Chapter 5.

3 Models

3.1 Show and Tell

3.1.1 Overview

This paper showed that we get State of the Art (SOTA) results when we directly maximize the probability of the correct description given the Image. Assuming that the Image is represented by I, θ denotes the parameters of our model and S is the correct transcription, we can find optimal parameters $\theta^{'}$ for the model such that:-

$$\theta^{'} = \underset{\theta}{\operatorname{argmax}} \sum_{(I,S)} log(p(S \mid I;\theta)) \tag{6}$$

To simplify the above expression, we can remove θ for convenience. Assuming that N is the length of this sentence:-

$$log(p(S \mid I;\theta)) = log(p(S|I)) = \sum_{t=0}^{i=N} log(p(S_t \mid I, S_0, S_1....S_{t-1}) \tag{7}$$

As you can see, the sum of log probabilities is optimized here over the training set. The probability of every word S_t, given that that were generated before it are $S_{t-1}, ...S_2, S_1$ is summed here.

The LSTM model described in Section 2.2 is used here. This model provides output m_t that goes through a softmax layer, and gives us p_{t+1} - a probability distribution over all words in the dictionary. The best next word is selected and used as part of the caption.

$$p_{t+1} = Softmax(m_t) \tag{8}$$

3.1.2 Model

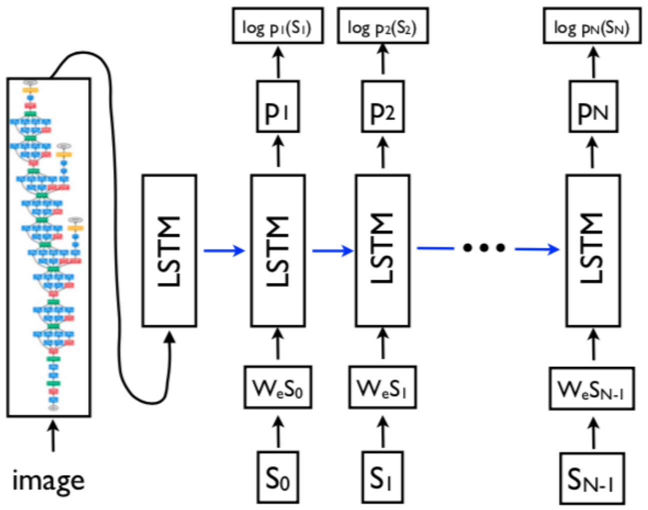


Figure 4: Show and Tell model

The Convolutional neural network used here is the GoogleNet. The LSTM predicts each word of the sentence from the Image Embedding, and it is useful to create a copy of the LSTM the image, for each sentence word. All LSTMs in the above image have same parameters, and output of one LSTM is the input of the next. Therefore, if we assume that I is the input image, and S=($S_0, S_1, ...S_N$) represents the caption where S_0 and S_N are represented by a special start and stop token:

CNN Embedding is the initial state of the first LSTM

$$x_{-1} = CNN(I) \tag{9}$$

Use same word Embedding W_e

$$x_t = W_e S_t, t \in 0, 1, 2..., N-1 \tag{10}$$

Probability is given by output of last LSTM

$$p_{t+1} = LSTM(x_t), t \in 0, 1, 2..., N-1 \tag{11}$$

We only use the Image I once here, and the word embedding W_e is the same for all t, thus mapping words and the image to the same space. The word embedding vectors here are independent of size of dictionary as opposed to a one hot embedding, where length of the word is the size of the dictionary and these can be jointly trained with the model.

3.1.3 Loss Function

Loss function used here is the negative log likelihood of correct word at each step and loss is minimized by using stochastic gradient descent with fixed learning rate, random weight initialization and no momentum.

$$L_{I,S} = -\sum_{t=1}^{N} log \ \ p_t(S_t) \tag{12}$$

3.1.4 Inference

As for approaches to generate a sentence, we can use either of the below:-

- Sampling:-
  First word is sampled according to output p_1, and the corresponding embedding is provided as input to sample p_2 and so on, recursively till output word is special stop token or maximum length of sentence is reached

- BeamSearch:-
  Iteratively consider best set B of size k, containing best sentences up to time t as candidates for generating sentences of size $t+1$ and continue process recursively. Beam search technique was used with beam size of 20 to approximate S as :-

$$S = \underset{S'}{\operatorname{argmax}}\, p(S^{'} \mid I), S' \in B \tag{13}$$

3.2 Show, Attend and Tell

3.2.1 Overview

This paper is based on the concept of attention to create a model that describes content of images. Here, the same encoder decoder framework used in Show and Tell is retained but with the attention framework, latent alignments are learnt from scratch to have the model attend to abstract concepts. Two variants of attention, namely Stochastic and Deterministic Attention are used here with difference in how the ϕ

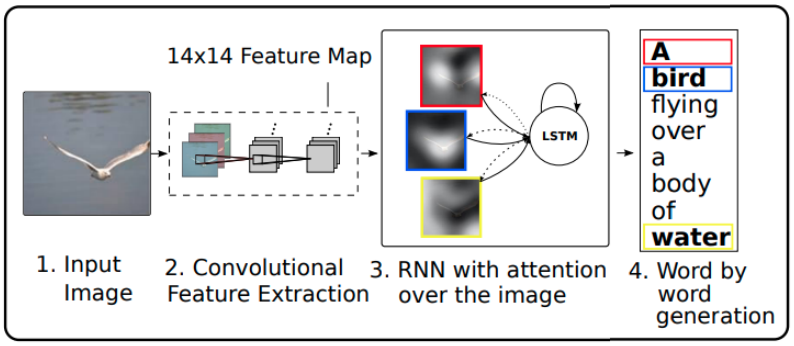

14x14 Feature Map
LSTM
A
bird
flying
over
a
body
of
water
1. Input Image
2. Convolutional Feature Extraction
3. RNN with attention over the image
4. Word by word generation

Figure 5: Show Attend and Tell model

function is defined.

Attention is a pluggable model that can be seamlessly inserted to remarkable improve caption quality. The concept of attention stems from the fact that nets considered every pixel in an Image as an input for the Encoder stage, and valued all of these equally. Attention mechanism broke this construct but selecting an arbitrary discrete portion of the image to use. This is analogous to the fact that we see images as a whole, our "attention" is focused on only a portion of the image.

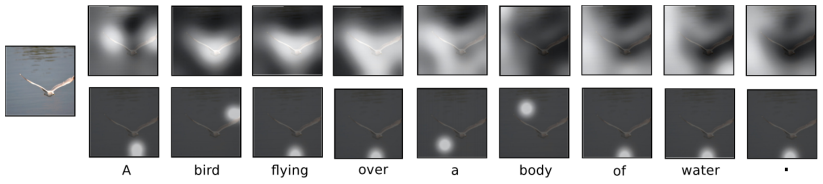


Figure 6: Hard (top) and Soft (bottom) Attention over time

3.2.2 Encoder

The Convolutional Encoder takes an image and generates a caption y encoded as a sequence of 1-of-K encoded words. If C is the length of the caption and K is the size of the vocabulary,

$$y = \{y_1, y_2 ..., y_c\}, y_i \in \mathbf{R}^K \tag{14}$$

We use a convolutional neural network to extract a set of L feature vectors, each of which is a D-dimensional representation corresponding to a part of the image and these are referred to as Annotation features. Features are extracted from a convolutional layer instead of a fully connected layer, to focus on parts of image and sub-select feature vectors.

$$a = \{a_1, a_2 ..., a_L\}, a_i \in \mathbf{R}^D \tag{15}$$

3.2.3 Decoder

The decoder used here is an LSTM too, but the notation and variation used is different.

Let's use $T_{(s,t)}$ to denote a simple affine transform. Assume i_t, f_t, o_t, c_t are input, forget, memory, output and hidden states of LSTM, vector $\hat{z_t} \in R^D$ as context vector and $E \in R^{mxk}$ as embedding matrix, where m is embedding dimension and n is LSTM dimension. σ and $\odot$ are logistic sigmoid and element-wise multiplication functions like before.

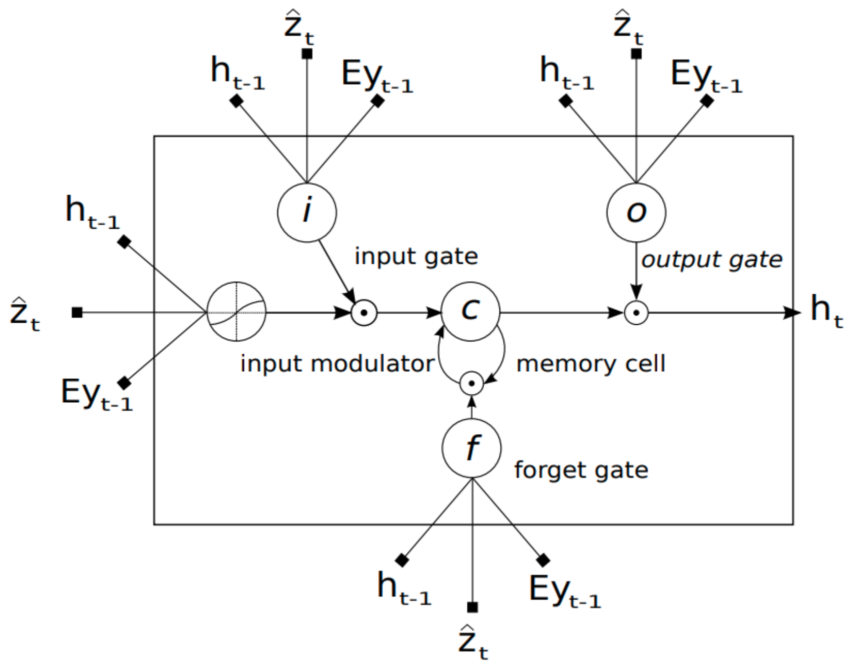


Figure 7: Show Attend and Tell model - LSTM

$$\begin{pmatrix} i_t \\ f_t \\ o_t \\ g_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{pmatrix} T_{D+m+n,n} \begin{pmatrix} Ey_{t-1} \\ h_{t-1} \\ \hat{z_t} \end{pmatrix} \tag{16}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{17}$$

$$h_t = o_t \odot tanh(c_t) \tag{18}$$

$\hat{z_t}$ is a representation of relevant part of input image at time t. We define an attention mechanism ϕ that computes $\hat{z_t}$ using a_i, where i=$1, 2, 3.....L$ using

$$\hat{z_t} = \phi(a_i, \alpha_i) \tag{19}$$

where α_i is the weight of each annotation vector a_i, computed by our attention model f_{att} . Also, the value of α_i and hidden state varies as words in the caption get generated as

$$e_{ti} = f_{att}(a_i, h_{t-1}) \tag{20}$$

3.3 Stochastic "Hard" Attention

Let s_t be the location in the image where the model focuses attention while generating the t^{th} word. Then, we can define $s_{t,i}$ as an indicator 1-hot variable such that,

$$s_{t,i} = \left\{ \begin{array}{ll} 1, & visual\ features\ extracted\ at\ i^{th}\ location \\ 0, & otherwise \end{array} \right\} \tag{21}$$

We can view $\hat{z_t}$ as random variable and assign a multinouilli distribution parameterized by α_i. and define :

$$p(s_{(t,i)} = 1 \mid s_{j<t}, a) = \alpha_{t,i} \tag{22}$$

$$\hat{z_t} = \sum_i s_{t,i} a_i \tag{23}$$

Using this, if we define L_s as a new objective function such that $L_s = log(p(y \mid a)$, and this function is a variational lower bound on marginal log likelihood of observing a sequence of words given image features. Using rule of chain rule and marginalization of probabilities, we can simplify L_s, and partially differentiate this with respect to model parameters to get the hard attention equation by maximizing the variational lower bound. This is equivalent to the REINFORCE learning rule, where reward for attention choosing actions is a real value proportional to log likelihood of target sentence. This is hence called hard attention, because we make a hard choice at every point, sampling a_i based on a multinouilli distribution parameterized by α.

3.4 Deterministic "Soft" Attention

Instead of sampling attention location s_t like in Hard Attention, we can take the expectation of context vector $\hat{z_t}$ directly as:-

$$E_{p(s_t|a)} \mid \hat{z_t} = \sum_{i=1}^{L} \alpha_{t,i} a_i \tag{24}$$

Soft Attention corresponds to feeding a soft α weighted context into the system. The model is smooth and differentiable so end-end learning is done using standard back-propagation. Here, we optimize the marginal likelihood under attention location random variable s_t. Overall, expectation of output can be induced by feedforward propagation with expected context vector $E[\hat{z_t}]$. This model predicts a scalar β from previous hidden state h_{t-1} at each time step such that $\phi a_i, \alpha_i = \beta \sum_i^L \alpha_i a_i$ where β_t is $\sigma(f_\beta(h_{t-1}))$. All the attention weights $\alpha_t i$ sum to 1 as they are the output of a softmax layer, and these are used to assign more importance to sections of the image. Negative log likelihood is used to train the model with a penalty factor here.

3.5 Training

Soft and Hard attention model are trained using stochastic gradient descent using adaptive learning rate algorithms.

4 Results

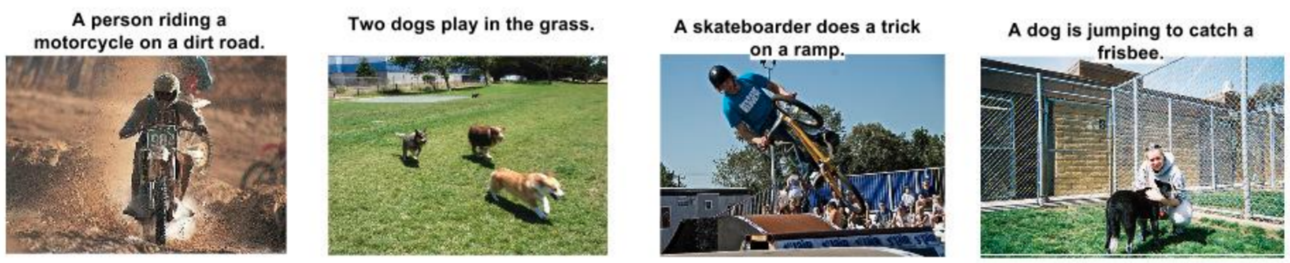

A person riding a motorcycle on a dirt road.
Two dogs play in the grass.
A skateboarder does a trick on a ramp.
A dog is jumping to catch a frisbee.

Figure 8: Show and Tell Results

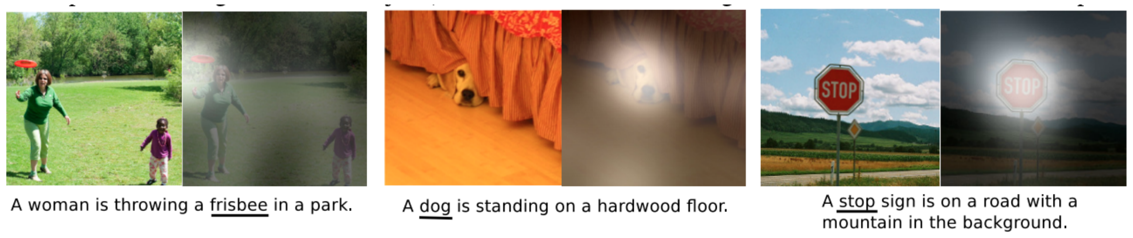

A woman is throwing a frisbee in a park.
A dog is standing on a hardwood floor.
A stop sign is on a road with a mountain in the background.

Figure 9: Show Attend and Tell Results

5 Dataset

Several datasets have been used for Image captioning and these consist of images and sentences in English describing these images. For this project, you are expected to use the MS COCO (Microsoft Common Objects in Context dataset). The MS COCO dataset is a large scale object detection, segmentation and captioning dataset and it has 5 captions per image. The images in this dataset have certain objects, colors, animals or people with distinguishing characteristics. You can access the dataset here.

6 Evaluation Metric

The most well-recognized evaluation metric for Image captioning is Human Evaluation. There are several conventional metrics such as BLEU, METEOR, ROUGE-L and CIDER and these have been used in Image captioning competitions to benchmark results along with human evaluation. Often machine translation metrics are used for image captioning evaluation, because we compare one/more than one caption against the generated caption.

BLEU score works by counting matching n-grams in candidate translation to n-grams in reference text. METEOR is based on the harmonic mean of unigram precision and recall between translation and reference text, with recall carrying a higher weight. ROUGE-L score works by using longest common subsequence in sequence n-grams to measure long matching sequence of words and provides an F-score using LCS-based precision and recall metrics. CIDER score is computed using average cosine similarity between sentences, which accounts for both precision and recall.

Evaluation can be done using the instructions and evaluation code, as detailed under this page. Ground truth captions and Image captioning model output captions can be compared using these metrics.

7 Guidelines

- You can pick any method of your choice by looking at the papers and implement it and try to get decent results.
- You can also make use of any pre-trained models and fine-tune them.
- After you get decent results by implementing an existing technique, you can try out any novel modifications in the method to get improved results to maximize your project grade
- Before selecting a method, please find out how long does it take for the network to train if you implement it.
- Towards the end of the quarter, the `DSMLP` cluster will become very busy, slow at times and there might be connectivity issues. Please keep these things in mind and start early and also explore other alternatives like google co-lab (12 hours free GPU) etc.

- You are encouraged to implement classes similar to ones introduced in Assignment 3 (`nntools.py`) to structure and manage your project. Make sure you use checkpoints to save your model after every epoch so as to easily resume training in case of any issues.

8 Deliverables

You will have to provide the following

1. **A 10 page final report**:
    - 10 pages **MAX** including figures, tables and bibliography.
    - One column, font size: 10 points minimum, **PDF format**.
    - Use of Latex highly recommended (e.g., NIPS template).
    - Quality of figures matter (*Graph without caption or legend is void.)*
    - The report should contain at least the following:
        - Introduction. What is the targeted task? What are the challenges?
        - Description of the method: algorithm, architecture, equations, etc.
        - Experimental setting: dataset, training parameters, validation and testing procedure (data split, evolution of loss with number of iterations etc.)
        - Results: figures, tables, comparisons, successful cases and failures.
        - Discussion: What did you learn? What were the difficulties? What could be improved?
        - Bibliography.
2. **Link to a Git repository** (such as GitHub, BitBucket, etc) containing at least:
    - Python codes (using Python 3). You can use PyTorch, TensorFlow, Keras, etc.
    - A jupyter notebook file to rerun the training (if any),

      → We will look at it but we will probably not run this code (running time is not restricted).
    - Jupyter notebook file for demonstration,

      → We will run this on UCSD DSMLP (running time 3min max).

      This is a demo that must produce at least one illustration showing how well your model solved the target task. For example, if your task is classification, this notebook can just load one single testing image, load the learned model, display the image, and print the predicted class label. This notebook does not have to reproduce all experiments/illustrations of the report. This does not have to evaluate your model on a large testing set.
    - As many jupyter notebook file(s) for whatever experiments (optional but recommended)

      → We will probably not run these codes, but we may (running time is not restricted).

      These notebooks can be used to reproduce any of the experiments described in the report, to evaluate your model on a large testing set, etc.
    - Data: learned networks, assets, ...(**5Gb max**)
    - **README file** describing:
        - the organization of the code (all of the above), and
        - if any packages need to be `pip` installed.
        - Example:

```
Description
===========
This is project FOO developed by team BAR composed of John Doe, ...

Requirements
============
Install package 'imageio' as follow:

        $ pip install --user imageio

Code organization
=================
demo.ipynb         --  Run a demo of our code (reproduces Figure 3 of our report)
train.ipynb        --  Run the training of our model (as described in Section 2)
attack.ipynb       --  Run the adversarial attack as described in Section 3
code/backprop.py   --  Module implementing backprop
code/visu.py       --  Module for visualizing our dataset
assets/model.dat   --  Our model trained as described in Section 4
```

9 Grading and submission

The grading policy and submission procedure will be detailed later.