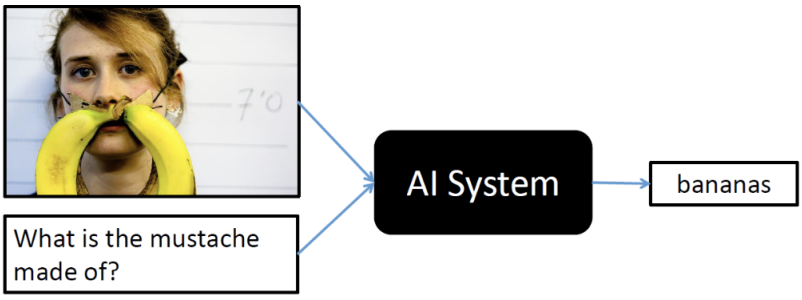
ECE 285 – MLIP – Project A
RNNs for Visual Question Answering

Written by Sneha Gupta. Last updated on October 18, 2019.

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 Description



Visual Question Answering is a research area about building a computer system to answer questions presented in an image and a natural language. VQA is an interesting challenge combining different disciplines, including computer vision, natural language processing, and deep learning. VQA represents not a single narrowly-defined problem (e.g., image classification) but rather a rich spectrum of semantic scene understanding problems and associated research directions. Much of this research, especially in the area of image classification, has been made possible by the publicly-available ImageNet database - which contains over four million images labeled with over a thousand object categories.

Example: Given an image and a free-form, natural language question about the image (e.g., "What kind of store is this?", "How many people are waiting in the queue?", "Is it safe to cross the street?"), the machine's task is to automatically produce a concise, accurate, free-form, natural language answer ("bakery", "5", "Yes").

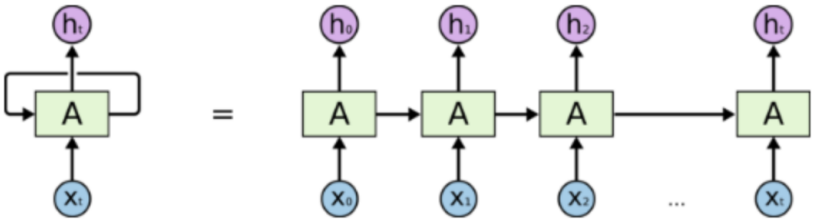
Please visit **official VQA site**, https://visualqa.org/, for the:

1. **Starter code**,
2. **Evaluation metric**,
3. **Dataset**[1].

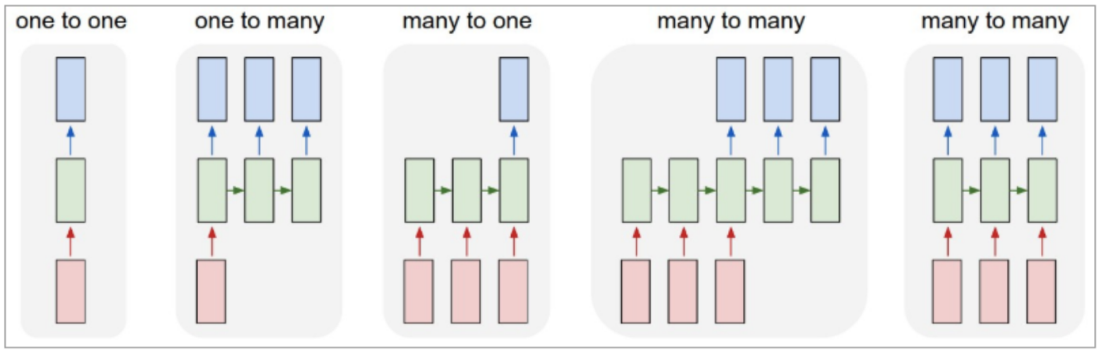
There are also some challenges and papers in the same area.

1.1 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.



An unrolled recurrent neural network.



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).

[1]The "balanced real images" dataset is available on DSMLP in `/datasets/ee285f-public/VQA2017/`.

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

Please look at some of the **relevant papers** in this area:
https://arxiv.org/pdf/1505.00468.pdf
https://github.com/yunjey/show-attend-and-tell
https://arxiv.org/pdf/1502.03044.pdf
https://arxiv.org/pdf/1511.02274.pdf
https://github.com/JamesChuanggg/awesome-vqa
https://arxiv.org/pdf/1805.09701.pdf

2 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.

3 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
```

4 Grading and submission

The grading policy and submission procedure will be detailed later.