TEK5040 Assignment, Image Captioning

Narada Warakagoda

September 24, 2023

NOTE: This assignment is optional. Therefore, no submission is required. But you are strongly encouraged to perform the tasks in this assignment

1 Introduction

Image captioning is a a term used in machine learning for the task of giving a textual description of an image. In this assignment, we will see how we can use RNNs (more specifically LSTMs) for this task. In addition, we will use a convolutional neural network to extract features from the image. The convolutional neural net- work will extract a feature map of size $7 \times 7 \times 1024$ for a an image of shape $224 \times 224 \times 3$. We will then learn a projection of the 1024-dimensional feature-vectors down to 256-dimensional vectors, mainly due to computational considerations. Furthermore we will use *soft attention* so that at each time step, we transform the $7 \times 7 \times 256$ feature map into a 256-dimensional context vector that is a weighted average of the 49 feature vectors. Note that when using content-based attention, the vectors are just treated as a flat array without having considerations to their spatial positions or relationships. If we want to use that kind of information we should use some spatial encoding scheme to embed positional information into the feature vectors themselves. But in this exercise no positional encoding is used. The sole idea is that at each time step we can attend to the location in the image that contains the relevant information for what we are currently describing. Figure 1 shows the architecture of the system implemented in this exercise.

1.1 Package Contents

`dataset.py`	Downloads and prepares data
`ed_model.py`	Defines encoder-decoder models
`losses.py`	Defines loss functions
`utils.py`	Defines dfunctions for evaluation and visualization
`train_test.py`	Defines train and validation steps
`main.py`	The main program
`exercise.pdf`	This document describing the task

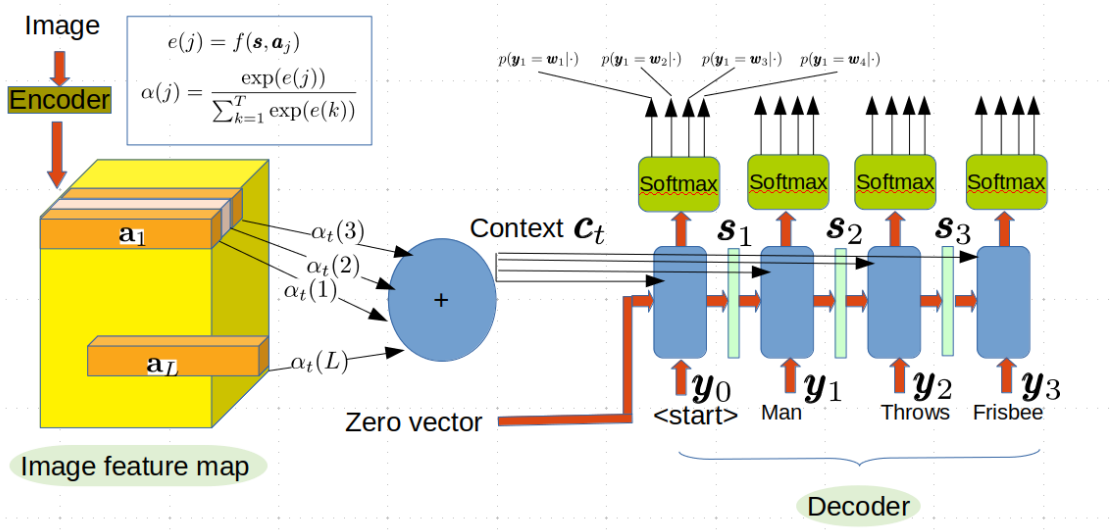

Figure 1: Image Captioning network

1.2 Operation

You can run the script by issuing the command

```
python3 main.py
```

at the command prompt in a Tensorflow environment. This will download a large data file (approximately 3GB) and trains the system for 10 epochs and test the system on a single image. Testing should print the generated caption. It also shows an array of images are the corresponding attention areas. Since the included code has not implemented a proper attention mechanism you should not be able to observe any clear attention areas. The main task of the exercise is to implement a proper attention mechanism.

If the system complains about missing python packages, install them. If you do not see the array of images, check you matplotlib configuration.

The code has been tested on Ubuntu 18 with Tensorflow 2.6.

2 Task

1. Study the implementation of the system. Most important files are `main.py`, `ed_model.py` and `train_test.py`.

2. Study the method `UniformAttention` found in `ed_model.py`. This method implements an attention that is uniform i.e. not a very useful attention mechanism.

   Attention is defined by the following two equations:

$$e_t(j) = f(s_t, a_j)$$
$$\alpha_t(j) = \frac{\exp(e_t(j))}{\sum_k \exp(e_t(k))}$$
$$c_t = \sum_j \alpha_t(j)\ a_j$$

where $f(\cdot)$ is some function that generates $e_t(j)$, whereas s_t is the current state vector, a_j is the j^{th} feature vector. Attention weights $\alpha_t(j)$ are used to calculate the context vector c_t.

In the case of uniform attention $e_t(j) = 0$ for any j and therefore it will return equal attention weights.

3. Implement the dot-product attention defined by the following function

$$e_t(j) = f(s_t, a_j) = (Qs_t)^T K a_j$$

where s_t is the current state vector, a_j is the j^{th} feature vector, whereas Q and K are trainable weight matrices.

You can just populate the template `DotProductAttention` in the file `ed_model.py`. Test your system, by uncommenting the line 44 in `main.py`.

4. Implement so called *Bahdanau* attention defined by the following function

$$e_t(j) = f(s_t, a_j) = W \tanh(Us_t + Va_j + b) + c$$

where s_t is the current state vector, a_j is the j^{th} feature vector, whereas W,U, V, b and c are trainable weight matrices/vectors.

You can just populate the template `BahdanauAttention` in the file `ed_model.py`. Test your system, by uncommenting the line 45 in `main.py`.

5. One can not in general say that there is a correct caption for a given image. Is supervised learning still an appropriate framework?