Gokul Karthik K (CS18SFP3156) gokulkarthikk@gmail.com
Guide: Prof. Harish G. Ramaswamy, Department of Computer Science and Engineering, IIT Madras
Period: 18 June, 2018 to 10 August, 2018

Summer Research Fellowship Program Report
Experimental Analysis of Attention Models in Deep Learning

Abstract:

Attention based neural network models have produced more accurate results in the recent years though there is no concrete theoretical proof on "Why attention works?". Hence we experimented the the models with and without attention under certain assumptions and tried to understand "How those models learn?".

Terminologies:

Artificial neural network:

An artificial neural network is the network of computation models which is loosely inspired from the biological neural network. It generally consists of an input layer, hidden layer(s) and an output layer. The data starts from the input layer and transformed by the operations of artificial neuron in the hidden layer(s) and output layer to produce the result.

In a supervised learning approach, the parameters associated with the layers are adjusted in each iteration of passing data such that it reduces the loss between the original and computed values.

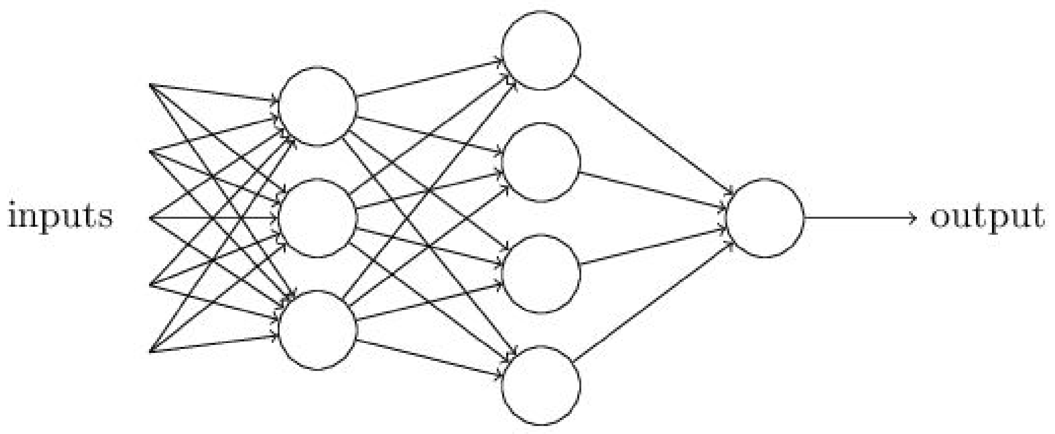


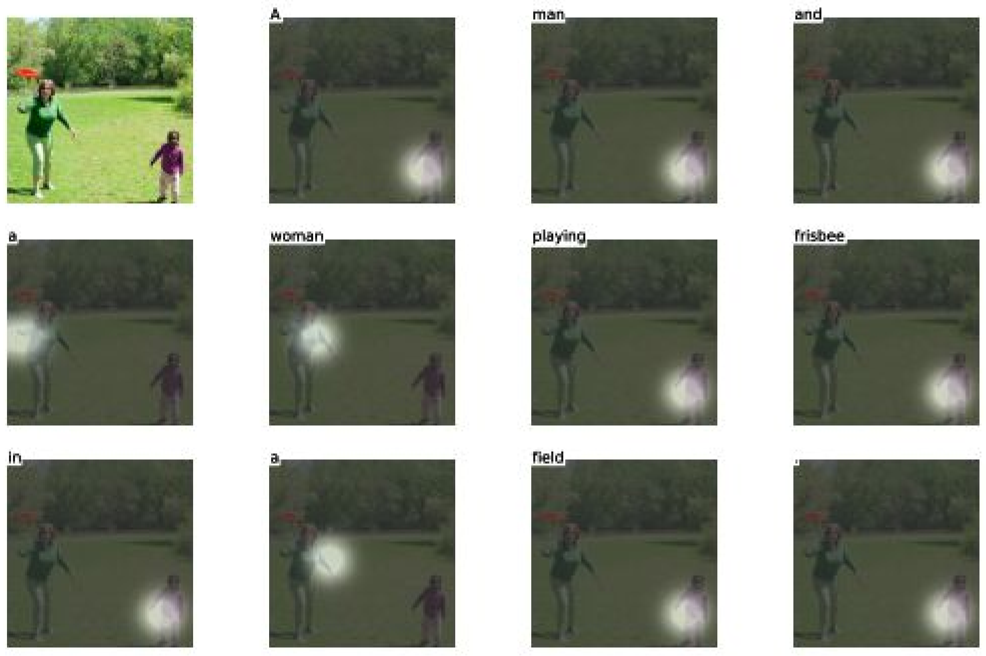
Fig. 1 Artificial Neural Network (Source: http://neuralnetworksanddeeplearning.com)

Attention model:

Attention is the mechanism that was developed to identify the focus area in the data. Before the attention mechanism in machine translation, the text data were transformed into a fixed length vector and machine learning algorithms were applied over that vector. But this truncation into a fixed size vector resulted in the loss of information. Attention mechanisms partially solves this problem by looking over all the information in the area and focusing on the specific area based on the context.

Attention for image captioning:

The mechanism of describing image similar to humans is called as image captioning. In a typical image captioning process, the encoder-decoder model is used. The image data are transformed into the vector space using convolutional neural networks in the encoder part. The recurrent neural network is then used over the encoded data to generate the caption. Attention mechanism when added in between the encoder and decoder to change the focus over time produced better results than the models without attention.



(a) A man and a woman playing frisbee in a field.

Fig. 2 Learning of image captioning model with attention

(Source: Show, attend and tell: Neural image caption generation with visual attention.)

Experiment:

Data Generation:

To understand what is happening in a controlled environment, we generated artificial data set with certain statistical properties for the foreground and background objects.

Let the image 'i' be segmented into 'k' parts where each part can be represented using a 'd' dimensional vector.

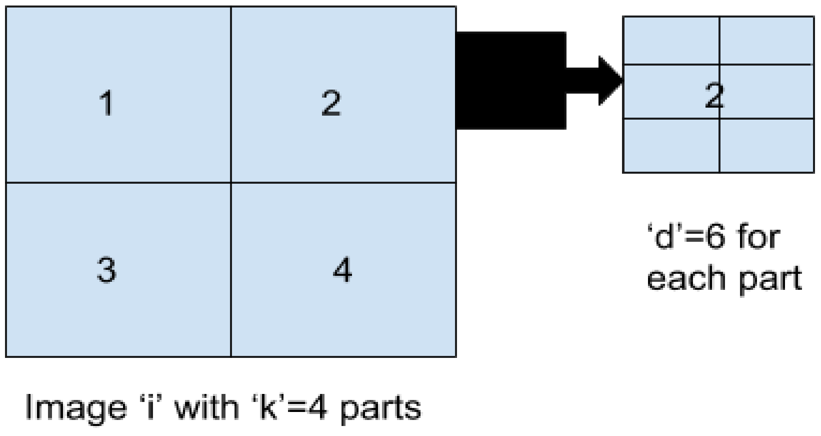


Fig. 3 Image representation

We assume that the foreground object to be in only one of these 'k' parts. Let that useful part be P_i and the other non useful parts be P'_i. We generated representation data for each image by the following algorithm:

```
Useful part = $i \in [1, k]$, selected uniformly at random

For each part in image 'i':

    If the part is useful:

        Generate data $\in R^d$ following the normal distribution with mean $\in R^d$ of 0s
        and variance $\in R^{dxd}$ of random values
```

```
Else:
    Generate data $\in R^d$ following the normal distribution with mean $\in R^d$ of 2s
    and variance $\in R^{dxd}$ of random values
```

Algorithm 1

Data classification:

Each data point consists of $k \times d$ values, in which only the part P_i is useful. Hence we used the $1 \times d$ values corresponding to the part P_i for classification. Let it be x_{useful}.

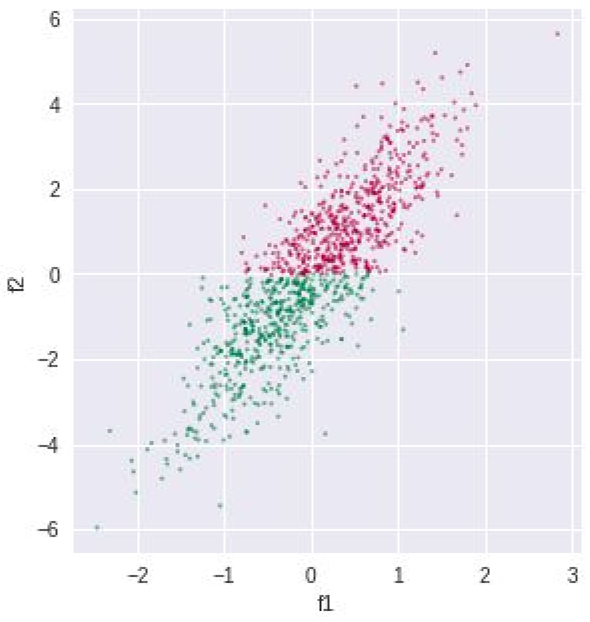


Fig. 4 Classification of data for ‘k’=25 and ‘d’=2, where ‘f1’ is the the first element of useful part P_i and ‘f2’ is the the second element of useful part P_i

Classification for each data point is done by:

$w \in R^{dx1}$ ***is generated uniformly at random with low=-1 and high=1***

$y = x_{useful} \times w$

```
If y < 0:
    class = -ve (0)
Else:
    class = +ve (1)
```

Algorithm 2

Machine learning models:

We built the following 3 types of models to classify the generated data. All those models can be generalized to have 3 hidden layers namely h_1 , h_{mid} and h_2 . Sigmoid is used as the activation function of h_{mid} in attention models. Let x be the data point and w^i is the weight matrix and b_i is the bias corresponding to each layer.

1. **Feed forward neural network without attention**

$$output = sigmoid(w^4(relu(w^3(relu(w^2(relu(w^1x + b_1)) + b_2)) + b_3)) + b_4)$$

2. **Feed forward neural network with attention and without aggregation**

$$output = sigmoid(w^4(relu(w^3(x \times softmax(relu(w^2(relu(w^1x + b_1)) + b_2))) + b_3)) + b_4)$$

3. **Feed forward neural network with attention and aggregation**

$$output = sigmoid(w^4(relu(w^3(sum(x \times softmax(relu(w^2(relu(w^1x + b_1)) + b_2))),\ axis = 1) + b_3)) + b_4)$$

Stochastic gradient descent is used as the optimizer and binary cross entropy is used as the classification loss measure. The number of epochs is fixed as 5000.

Layer	Weight	Model 1		Model 2		Model 3	
		Order	**Activation**	**Order**	**Activation**	**Order**	**Activation**
h_1	w^1	$(k \times d) \times h_{1n}$	Relu	$(k \times d) \times h_{1n}$	Relu	$(k \times d) \times h_{1n}$	Relu
h_{mid}	w^2	$h_{1n} \times h_{midn}$	Relu	$h_{1n} \times h_{midn}$	Relu, Softmax	$h_{1n} \times h_{midn}$	Relu, Softmax
h_2	w^3	$h_{midn} \times h_{2n}$	Relu	$(k \times d) \times h_{2n}$	Relu	$d \times h_{2n}$	Relu
$output$	w^4	$h_{2n} \times 1$	Sigmoid	$h_{2n} \times 1$	Sigmoid	$h_{2n} \times 1$	Sigmoid

Table 1 Model specifications

Attention Loss:

We modelled attention loss as the log loss between the TRUTH vector $\in \{0, 1\}^k$ denoting the usefulness of each part of a data point in classification and the result of softmax applied to the output of hidden layer h_{mid} .

Tools and compute:

Programming: Python 3

Deep Learning Library: PyTorch

Other Libraries: Numpy, Matplotlib, Scikit-Learn, JSON

Compute: Google Colaboratory with K80 GPU (Thanks to Google :-))

Code: https://github.com/GokulKarthik/Attention-Experiments-DL

Formatting the results:

The meta data and the results of each experiment are organised in JSON for further analysis. Thus a list of JSON objects where each element corresponding to each experiment is generated. The binary cross entropy loss, prediction accuracy and attention accuracy are noted once in every 50 iterations for storage space optimisation.

Key Insights:

py> pass

References:

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In the Annual Conference on Neural Information Processing Systems (NIPS)

[2] Xu, K., Ba, J. L., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R. S., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In the International Conference on Machine Learning (ICML).

[3] Bahdanau, D., Cho, K., Bengio, Y. (2015). Neural Machine Translation by Jointly Learning to Align and Translate. In the International Conference on Learning Theory (ICLR).