**CS9660B Project Report - Twitter Sentiment Analysis**

Arnab Mallik

Western University, Canada

[amallik3@uwo.ca](mailto:amallik3@uwo.ca)

1. **Abstract**

The main focus of the project is to analyze sentiment on Twitter dataset. Three deep learning methods have been trained to classify tweets based on their perceived sentiment, which will segregate the tweets as positive tweets or negative tweets. Experimental evaluations were applied to gauge the performance of each classification model and compare them. The best model was able to achieve an accuracy of 0.8358 on Kaggle leaderboard.

1. **Description of Applied Problem**

Twitter is one of the most popular social media platforms where people post short messages about their life, share opinions concerning different topics and discuss current issues [1]. This is backed up by the fact that there are more than 336 million active users and more than 500 million tweets are generated every day.

Twitter members interact with each other using messages known as tweets. Tweets can be efficiently used for observing market intelligence, public sentiment regarding political movement, or people’s attitudes on current social debates [2].

According to the Cambridge dictionary, ‘Sentiment’ is defined as a thought, opinion, or idea based on a feeling about a situation, or a way of thinking about something. Sentiment is a belief that has emerged from our emotions and they are generally shared by groups of people.



Fig 1: Crux of the project is to classify tweets into 2 classes

Political parties may want to know whether people support their program / campaign or not. Many social public movements such as “#MeToo” movements started and became wide spread because Twitter. A company may want to find out the reviews of its products. Therefore, building models to classify a tweet as expressing a positive or negative sentiment is vital and meaningful.

In this project, dataset from Kaggle [3], which was obtained by crawling Twitter and the labels were manually hand labelled as positive (1) or negative (0). The dataset contains emoticons, usernames and hashtags which are required to be cleaned and tweets converted into a standard form. We also need to extract useful features from the text and encode them as feature vectors.

1. **Description of Data Set**

The dataset is in CSV format. The train CSV contains 3 columns: ‘tweet\_id’, ‘tweet’ and ‘sentiment’.

The column ‘tweet\_id’ is just index number of the tweets. ‘tweet’ is the actual tweet message and ‘sentiment’ defines the sentiment of the tweet and is either negative (0) or positive (1). The test dataset only includes columns ‘tweet\_id’ and ‘tweet’. The training dataset contains 100,000 records. The test dataset contains 30,000 records. During training each model, 10% of training data is randomly selected as validation set. Remaining 90% is used for training. Fig 2 shows some of the key stats about the training data set.

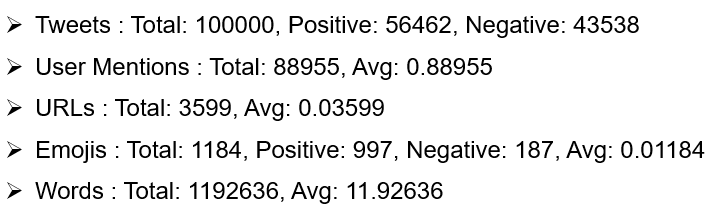


Fig 2: Stats about training data

1. **Methodology and Implementation**

Fig 3 illustrates the important steps required for implementing classifier models.

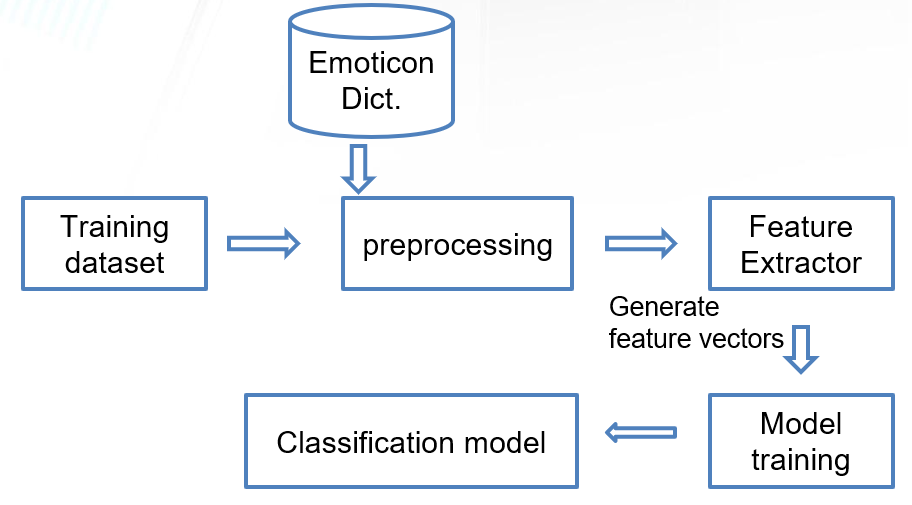
****

Fig 3: Training Process Flow

* 1. **Pre-Processing**

Raw data scraped from twitter will result in a noisy dataset. Thus, both the training and test dataset will have noise in them. This is because of the casual nature of social media usage. There are some special characteristics of tweets such as retweets, emoticons, user mentions, etc. which must be suitably cleaned. The pre-processing performed on tweets are as follows.

* Convert tweets to lower case
* Replace punctuations with space
* Replace multiple white spaces with a single white space
* Replace any URL present in tweets with the word “URL”
* Replace any @handle with the word “USER\_MENTION”
* Replace all emotions with either “EMO\_POS” or “EMO\_NEG” according to Fig 4
* Replace all # (present in hashtags) with whitespace
* All retweets begin with RT. Strip the RT part.

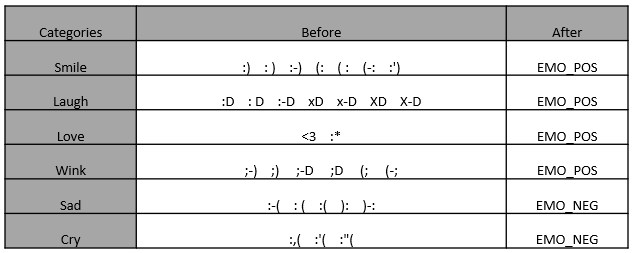


Fig 4: Preprocess Emoticons

* 1. **Feature Extraction**

Each Tweet Text is parsed from the CSV and split into words. These words are then added into a word list and a frequency distribution is created on the unigrams. Fig 5 shows the most frequently appearing unigrams. The frequency distribution is required to provide ranking to the unigrams based on their importance in the corpus.

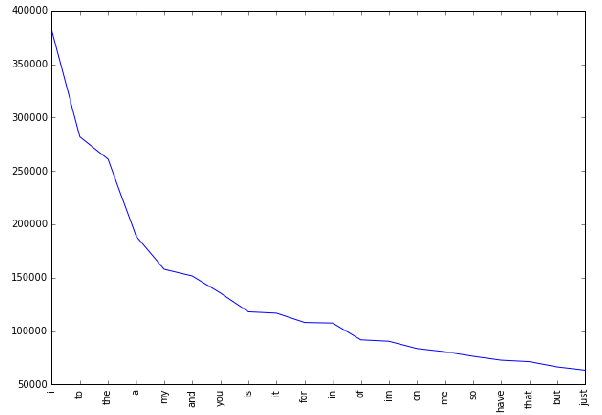
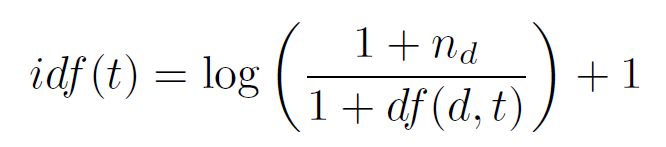


Fig 5:Frequencies of top 20 unigrams

* 1. **Feature Representation**

After each tweet is extracted, all the Tweets are represented as feature vectors as either a Sparse Vector or Dense Vector.

A total of 181232 unique words were extracted from the training dataset. All these words are plotted in a frequency distribution and the 15,000 highest ranked words are considered for Feature Vector. The remaining words have been considered as noise and hence been excluded. Thus, our corpus known 15000 words (only when considering Sparse Representation) and each tweet is represented as a vector of length 15000. Each unigram is given a unique rank based on its importance. The importance of each word is calculated using *tf-idf* [7].



Thus, for each tweet, the feature vector will have a value of 0 for words not present in tweet. For words that are present, it will have some positive value based on the *tf-idf* value of that corresponding word.

In case of Dense Representation, the vocabulary size is of 90,000. Pre-trained GLOVE word embeddings have been used [6]. Thus, each word is a vector of length 200.

1. **Experiments**

Experiments were performed using 3 models. The models are: Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM). In case of MLP, the sparse vector representation was used. For the other two models, dense vectors were used.

For all 3 models, 10% of the training data (randomly chosen) was used as Validation Set. The remaining data was used for training the models.

* 1. **Multilayer Perceptron (MLP):**

Multilayer Perceptron or MLP is a feed-forward neural network, with a minimum of three

layers of neurons. Each layer of neuron applies a non-linear activation function to the weights and these weights are learnt through supervised training using backpropagation. Training the MLP implies the learning of the weights between each layer such that the cost is minimum [8]. MLP performs well in difficult problems like sentiment analysis by learning non-linear models. Figure 6 shows the architecture of MLP used in project.

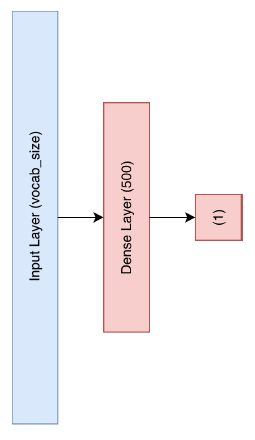
****

Fig 6:Architecture of MLP used

The MLP used in project contains 3 layers – Input layer with Input layer size of 15000. Hidden layer with 500 neurons and an output layer with 1 neuron. The loss function used is Binary Cross Entropy and the Accuracy is used as the performance measure. Adam optimizer is used for rapid convergence to a minimum cost.

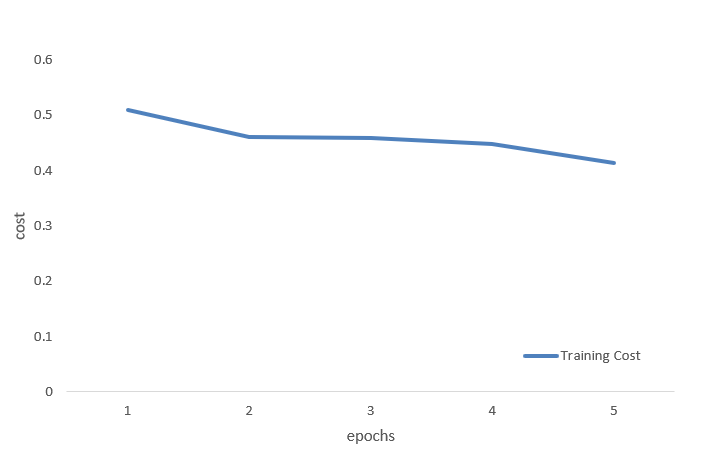
****

Fig 7:MLP training cost Vs epochs

Fig 8 shows the training accuracy and the validation accuracy over 5 epochs. The model starts to overfit from epoch 3. Thus, the best model is chosen to be the model at epoch 2. In the Test set, the best model achieves an accuracy of 76.75%.

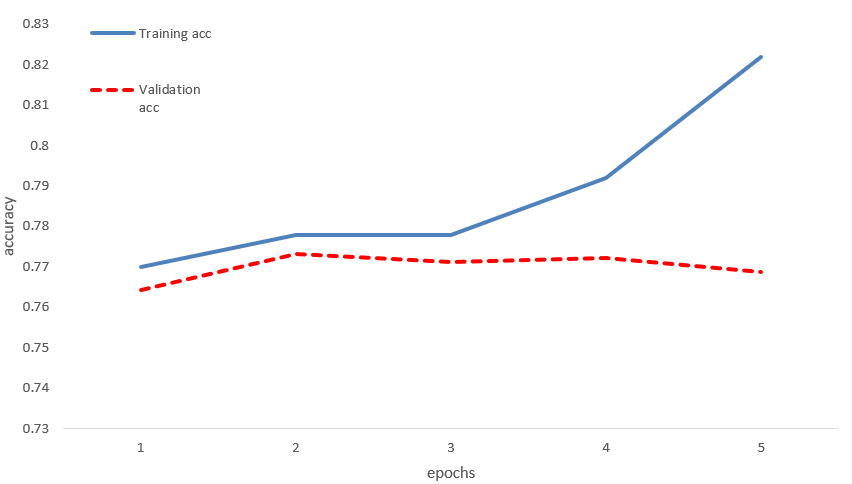
****

Fig 8:MLP accuracy Vs epochs

* 1. **Convoluted Neural Network (CNN):**

Convolutional Neural Networks or CNNs are a type of neural networks where layers called convolution layers are used. The unique properties of these layers help to interpret spatial data. A convolution layers also uses multiple filters or kernels, using which it learns to extract specific types of features from the data. During the training phase, the values of the filters are learnt over time. The kernel is a 2D window which is slides over the input data performing the convolution operation [4]. Sometimes Pooling layers are used to improve the performance of network. In the project, temporal convolution has been used which is ideal for analyzing NLP problems.

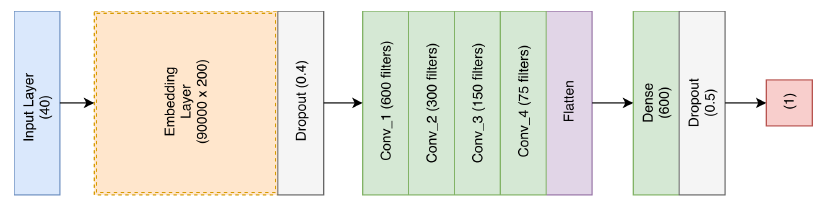
****

Fig 9:Architecture of CNN used

Fig 9 and Fig 10 shows the architecture of the CNN used. The Glove embeddings are of length 200. The longest tweet present in the dataset is of length 40. Thus, each tweet is represented as a matrix of 40 x 200. If the tweet has less than 40 words, then padding is added at the end. Dropout layers are used which is used for regularization and increases performance of the network. 4 convoluted layers are used, and all layers use relu activation function. Only the last layer uses sigmoid function as the classifier is binary. The vocabulary size used for this model is 90,000.

Binary cross entropy was used as loss function and accuracy as the performance metric. Adam optimizer is used.

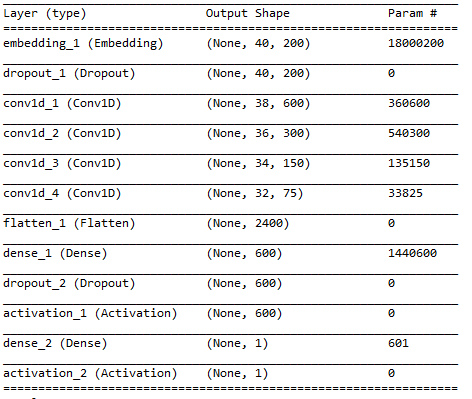
****

Fig 10:Architecture of CNN used

Fig 11 shows the training cost and validation cost over 8 epochs. Training cost decreases over each successive epoch but validation cost increases after epoch 3.

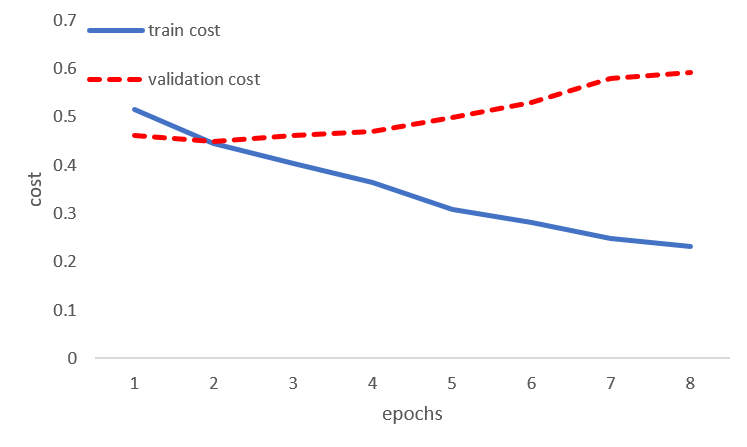
****

Fig 11:CNN training cost Vs epochs

Fig 12 shows the training accuracy and the validation accuracy over 8 epochs. The model starts overfitting from epoch 3, as the training accuracy increases but the validation accuracy decreases. The best model is the one at 3rd epoch and it provides an accuracy of 83.24% on the test dataset.

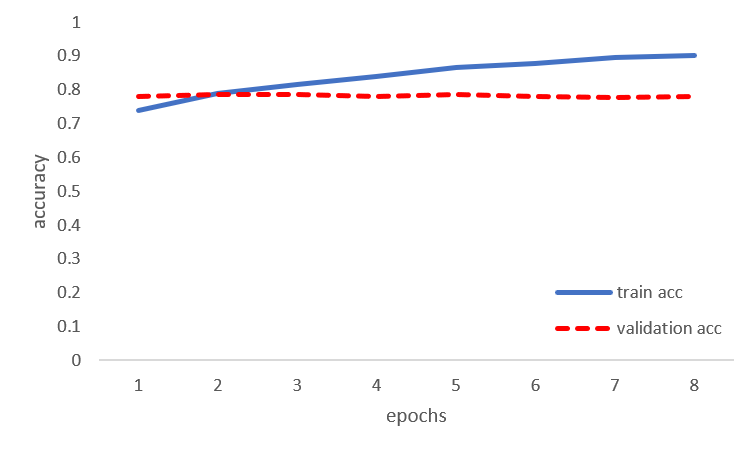
****

Fig 12:CNN accuracy Vs epochs

* 1. **Long Short-Term Memory (LSTM):**

Long Short-Term Memory networks – usually just called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by [Hochreiter & Schmidhuber (1997)](http://www.bioinf.jku.at/publications/older/2604.pdf), and were refined and popularized later by others. They work tremendously well on a large variety of problems, and are now widely used.

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn.

All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer [9].

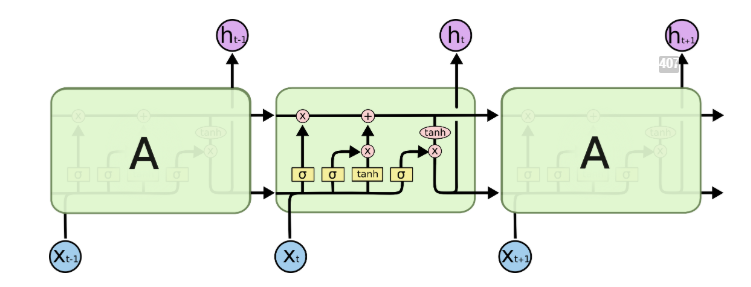


Fig 13:Repeating module in LSTM

Fig 14. Shows the architecture of the LSTM used in the project. Embedding layer is of size 40 x 200, reason is same as CNN. Vocabulary size of 90,000 is considered. Relu activation function is used in all the layers except the output layer, which uses sigmoid function. Loss function of binary cross entropy is used. Accuracy is used as the performance measure. Adam optimizer is used.

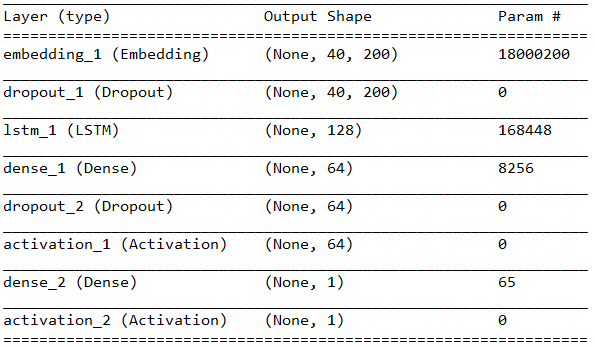
****

Fig 14:Architecture of LSTM used

Fig 15 shows the training cost and validation cost for 5 epochs. The training cost keeps on decreasing, whereas, the validation cost starts to increase after epoch 4.

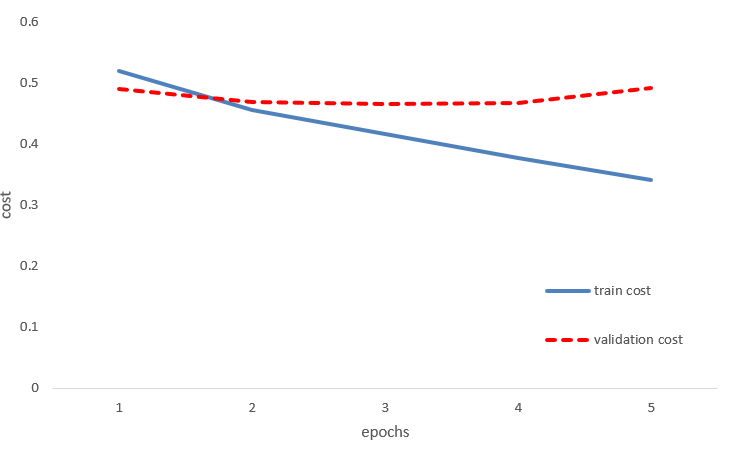
****

Fig 15:LSTM training cost Vs epochs

Fig 16 shows the training accuracy and validation accuracy over 5 epochs. Overfitting occurs after epoch 3, thus model obtained at epoch 3 is the best model. The accuracy of the best model on Test data set is 83.58%.

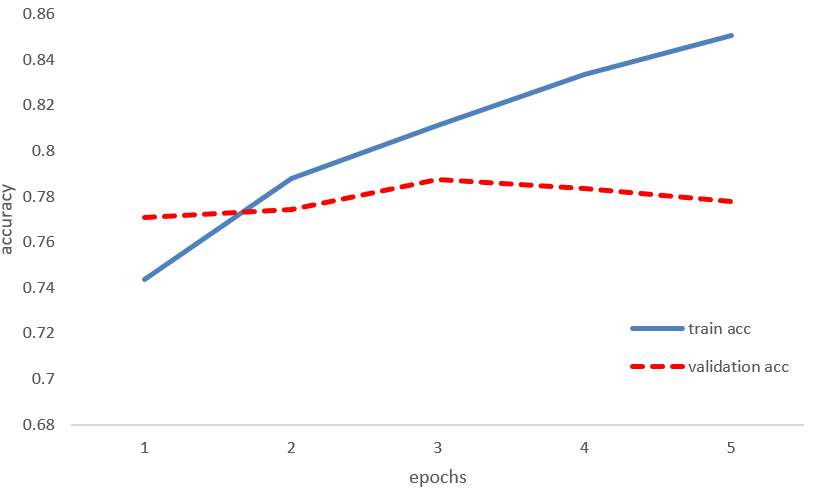
****

Fig 16:LSTM accuracy Vs epochs

1. **Conclusion**

The tweets present in the dataset had words, emoticons, hashtags, URLs, user handles and other characters. These tweets had to be cleaned in pre-processing step before training the models. Three models were trained – MLP, CNN, LSTM. Sparse feature vector was used for MLP, dense feature vector was used for CNN and LSTM. For all three models, binary cross entropy was used as the cost function and accuracy was used as performance matrix.

All three models performed well, where the best accuracy of 83.58% was obtained by using LSTM.

1. **Future Works**

* **Neutral Sentiment:** Some tweets convey facts and do not have any sentiment associated with them. They must be classified as Neutral Sentiment and not as Positive or Negative. The project should be augmented to allow Neutral Sentiment.
* **More Classifier:** For the project, many more classifiers must be tried out to see which gives better performance. Classifiers like SVM, Decision Tree, Random Forest must be tried [5].
* **Stemmer:** Use of stemmers like Porter Stemmer, Krovetz Stemmer could increase the performance drastically. Stemmers could be used in future versions.

1. **References**

[1] S. Kiritchenko, X. Zhu and S. Mohammad, "Sentiment Analysis of Short Informal Texts", Journal of Artificial Intelligence Research, vol. 50, pp. 723-762, 2014.

[2] V. Patodkar and S. I.R, "Twitter as a Corpus for Sentiment Analysis and Opinion Mining", IJARCCE, vol. 5, no. 12, pp. 320-322, 2016.

[3] https://www.kaggle.com/c/6740/

[4] Y. Kim. Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882, 2014.

[5] T. Mitchell. Machine Learning. McGraw-Hill, 1997

[6] https://nlp.stanford.edu/projects/glove/

[7] H. Wu and R. Luk and K. Wong and K. Kwok. "Interpreting TF-IDF term weights as making relevance decisions". ACM Transactions on Information Systems, 26 (3). 2008.

[9] M. W. Gardner, S. R. Dorling et. al., “Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences”, Atmospheric Environment, Vol 32, Issue 14-15, 2627-2636.

[8] https://colah.github.io/posts/2015-08-Understanding-LSTMs/#fn1