A Study of Deep Learning on Sentence Classification

By: Trinh Hoang-Trieu
Supervisor: Nguyen Le-Minh
Nguyen lab, School of Information Science
Japan Advanced Institute of Science and Technology
thtrieu@apcs.vn

Abstract

The author study different deep learning models on the task of sentence classification. Dataset used is TREC. Models under investigation are: Convolutional Neural Networks proposed by Yoon Kim, Long Short Term Memory, Long Short Term Memory on top of a Convolutional layer, and Convolutional network augmented with max-pooling position. Results show that architectures with time-retaining mechanism works better than those do not. The author also propose new changes to the conventional final layer, namely to replace the last fully connected layer with a Linear Support Vector Machine, or to replace the discriminative nature of the last layer with a generative one.

1. TREC dataset

TREC dataset contains 5500 labeled questions in training set and another 500 for test set. The dataset has 6 labels, 50 level-2 labels. Average length of each sentence is 10, vocabulary size of 8700. A sample from TREC is as follow:

```
DESC:manner How did serfdom develop in and then leave Russia ?

ENTY:cremat What films featured the character Popeye Doyle ?

DESC:manner How can I find a list of celebrities ' real names ?

ENTY:animal What fowl grabs the spotlight after the Chinese Year of the Monkey ?

ABBR:exp What is the full form of .com ?

HUM:ind What contemptible scoundrel stole the cork from my lunch ?
```

We also experiment on the Vietnamese TREC dataset, which is a direct translation of the TREC dataset and thus, also have very similar statistics.

2. Word2Vec dataset

In this experiment, we also used another dataset to support training. The dataset consists of pre-trained 300-dimension embeddings of 3 millions words and phrases [1]. These embeddings will be used as the representation of input for training. This covers 7500 out of 8700 words present in the TREC vocabulary.

For the Vietnamese TREC dataset, we used another pre-trained dataset, which covers 4600 words out of approximately 8700 words in the Vietnamese TREC dataset.

3. Convolutional Neural Networks for Sentence Classification

Yoon Kim proposes a simple yet effective convolutional architecture that achieves remarkable result on several datasets [2]. The network consists of three layers. The first uses 300 convolutional filters with varying size to detect 300 different features across the sentence's length. The second performs maximum operation to summarise the previous detection. The last is a fully connected layer with softmax output.

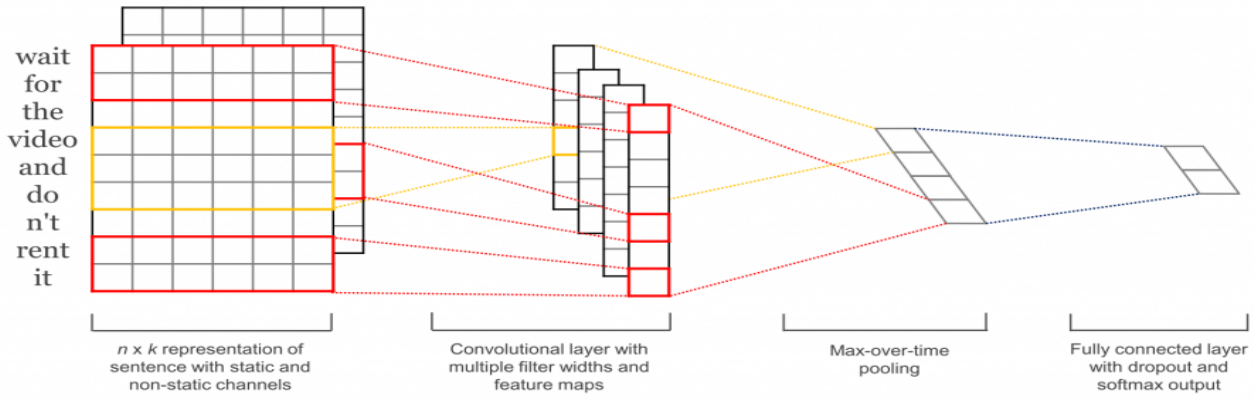


Figure 1. Convolutional neural network for sentence classification

The details ofS this network is as follow:

- Layer 1: 3 types of window size 3x300, 4x300, 5x300. 100 feature maps each.
- Layer 2: max-pool-over-time.
- Layer 3: Fully connected with softmax output. Weight norm constrained by 3.
    - Dropout with p = 0.5

4. First experiment: Long Short Term Memory

The authors first experiment with a simple Long Short Term Memory architecture. We used the many-to-one scheme to extract features before discriminating on these extracted features by a fully-connected layer.

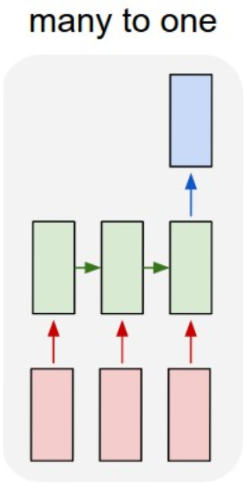


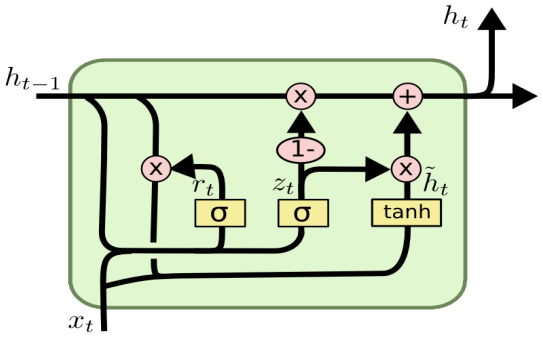
Figure 2. Many-to-one structure.
Source:
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

The detail of this network is as follow:

- First Layer: Word Embedding
- Second Layer: Long Short Term Memory
- Third Layer: Fully connected with Softmax output

For the second layer, we experimented on three different variations of Long Short Term Memory Cell:

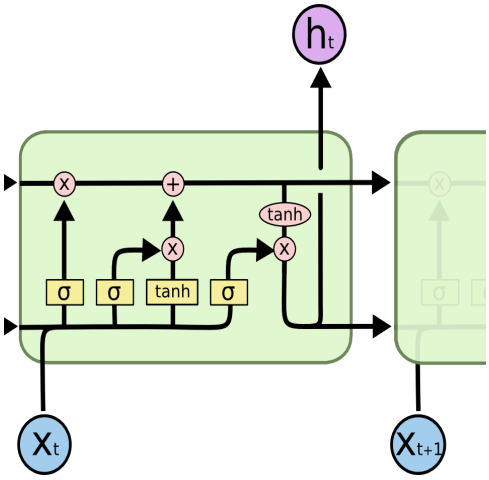
4.a. Gated Recurrent Unit



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 3: Gated Recurrent Unit
Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

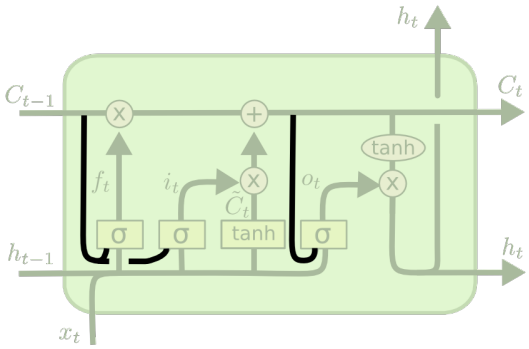
4.b. Basic Long Short Term Memory



$$a_t = [h_{t-1}, x_t]$$
$$f_t = \sigma(W_f \cdot a + b_f)$$
$$i_t = \sigma(W_i \cdot a + b_i)$$
$$C'_t = tanh(W_C \cdot a + b_C)$$
$$C_t = f_t * C_{t-1} + i_t * C'_t$$
$$o_t = \sigma(W_o \cdot a + b_o)$$
$$h_t = o_t * tanh(C_t)$$

Figure 4: Basic Long Short Term Memory
Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

4.c. Long Short Term Memory with peepholes



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

Figure 5: Long Short Term Memory with peepholes
Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

5. Long Short Term Memory on top of a Convolutional layer

Instead of max-pool-over-time, the author propose a softer version where the maximum operation is perform over a smaller region of the feature maps. By this, temporal information is better retained, max-pool layer does not produce a single value but a sequence instead. After this layer, the information is then feed into a Long Short Term Memory Cell with many-to-one scheme and a fully connected layer with softmax output. The authors also experimented with three aforementioned types of LSTM Cell.

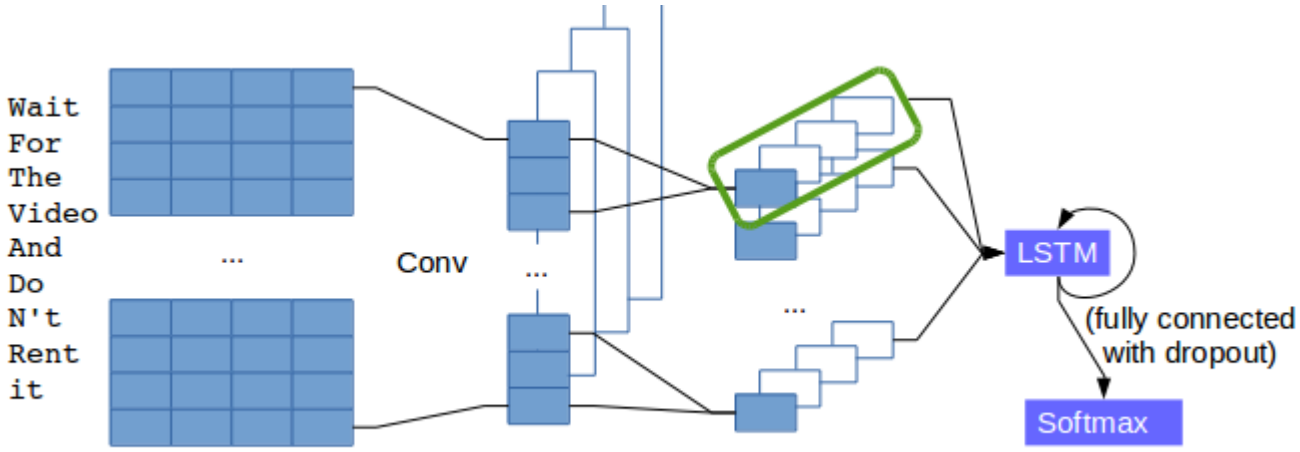


Figure 6: Long Short Term Memory on top of a Convolutional Layer

6. Convolutional network augmented with max-pooling position

In this network, besides performing maximum operation at the max-pooling layer, we also apply the argmax function to produce the position in which this maximum value takes place. By supplying this position information, the authors expect a better performance since this is also a time-retaining mechanism, only without any recurrent connection.

To be able to propagate gradient information through this argmax operation, the authors approximate this argmax operation with another differentiable function:

$$argmax(x) \approx [1\,2\,3 \ldots L]^T softmax_{\alpha \to \infty}(\alpha x)$$

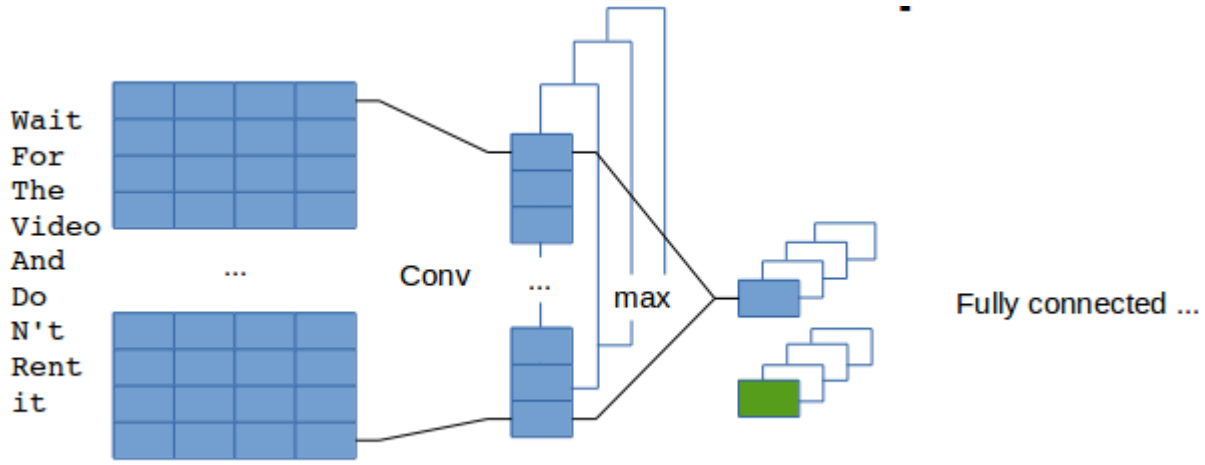


Figure 7: Convolutional network with max-pooling position

7. Hyper-parameter tuning and Experimental details

The authors use 10-fold cross validation on training set for hyper parameter tuning. The word embedding input is kept static throughout all model except the first one to replicate Yoon Kim's experiment. Since the average length of the sentence is only 10, while the maximum length of sentences in the dataset is approximately 40, we also did not use paddings to avoid feeding the networks too much noise (otherwise, on average there will be 30 padding words for each 10 words in the dataset).

Words that do not appear in the pre-train embedding dataset is initialised with variance 0.25 to match with existing words. For training procedure,we hold-out 5% of the data for early stopping since the dataset is not too big. Training is carried out in batches of size 50 to speed up convergence.

8. Result

The LSTM Cell with peepholes gives best result when a single recurrent layer is used (described in Section 4), while Gated Recurrent Unit gives best result in the hybrid model of Convolution and Recurrent connections (described in Section 5).

Performance of the Convolutional Network augmented with position is very unstable due to the presence of argmax operator. In our implementation, argmax is approximated with a function involves very big constant, which in turn results in unreliable gradients and poor result. In our model, the final layer is a fully connected layer, which is also not the best choice since classification is unlikely a linear function of features and their position. Other results are reported as follow:

	TREC	**TRECvn**
CNN (implemented on our system)	93	91.8
CNN-LSTM	94.2	92.8
LSTM	95.4	94.2

Table 1. Accuracy (%) of different models on two datasets

9. Conclusion

As can be seen, models which utilises temporal information gives better results comparing to those do not.

10. Future work: Support Vector Machine as the final layer

Of all the proposed model, the final layer is always a fully connected layer with softmax output, which is equivalent to a Linear Classifier. In this case, the previous layers act as a feature extractor. Good performance as reported indicates that these feature extractor is able to extract useful features that represent points that can be separated well by a simple Linear Classifier. It is highly likely that these features are also useful for other kind of Linear Classifier.

The authors propose using Linear Support Vector Machine as the final layer, this can be done by simply replacing the usual cross-entropy loss function by Linear Support Vector Machine's loss function. Namely, instead of using:

$$Loss = \texttt{CrossEntropy}(\boldsymbol{target}, \mathbf{softmax}(W\phi(input)))$$

(Where $\phi(input)$ is the extracted features from the previous layers)

With Support Vector Machine being the final layer, loss function is now:

$$Loss = \sum_{i}^{n}\sum_{j}^{6} RELU(-t_j W_j \phi_i + 1) + C|W_j|^2$$

Where *RELU* stands for Rectifier Linear Unit function, *t* is the target that is either -1 or 1 . This loss function represent the loss when one-vs-rest scheme is used (since TREC is a multiclass dataset). For Linear Support Vector with a soft-margin, the loss function is:

$$Loss = \sum_{i}^{n}\sum_{j}^{6} RELU(-t_j W_j \phi_i + 1 - \xi_{ij}) + RELU(-\xi_{ij}) + C|W_j|^2$$

11. Reference:

[1] Kim, Yoon. "Convolutional neural networks for sentence classification." *arXiv preprint arXiv:1408.5882* (2014).

[2] Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." *Advances in neural information processing systems*. 2013.