COMP 551 MiniProject 4: Reproducible Machine Learning

T1-04: Convolutional Neural Networks for Sentence Classification

Zirui Kuai
ID: 260614908
zirui.kuai@mail.mcgill.ca

Kaylee Zhao
ID: 260719382
qifei.zhao@mail.mcgill.ca

Yihe Zhang
ID: 260738383
yihe.zhang@mail.mcgill.ca

April 18, 2019

Abstract

This work reproduces the baseline CNN model proposed by Yoon Kim [Kim, 2014] and improves the performance of the model by tuning hyperparameters and slightly modifying the architecture of the model. In addition, simpler algorithms with less computational complexity such as Naive Bayes Support Vector Machine (NBSVM) [Wang and Manning, 2012] are also explored, to compare with the baseline model mentioned in Kim's work. We discovered that the simpler algorithms may outperform the deep learning models in terms of both accuracy and computational cost.

1 Introduction

Deep learning models have been applied to solve computer vision [Krizhevsky et al., 2012] and natural language processing [Graves et al., 2013] problems. Originally, Convolutional Neural Networks were mainly applied in the field of image processing [Krizhevsky et al., 2012], while language processing tasks were solved by using Recurrent Neural Networks [Graves et al., 2013]. In recent years, CNN models have been proved to be effective for natural language processing (NLP) tasks, such as semantic parsing [Yih et al., 2014], search query retrieval [Shen et al., 2014], sentence modelling [Kalchbrenner et al., 2014] and other traditional NLP tasks [Collobert et al., 2011]. Furthermore, Kim [Kim, 2014] proposed to exploit the application of CNN on sentence classification, with simple model architectures.

In this work, we attempt to reproduce the baseline models mentioned by Kim, including the CNN model without pre-trained word embedding (CNN-rand), and the Naive Bayes Support Vector Machine (NBSVM) model, as well as to explore a modified model with better performance for the same task.

2 Related Work

The Convolutional Neural Network (CNN) is designed to take advantage of the 2D structure of an input image previously [Krizhevsky et al., 2012]. However, in recent studies [Johnson and Zhang, 2014], it is suggested that using only 1D structure (word order) of text data for CNN would also result a feasible prediction. In the mentioned work, Johnson and Zhang directly applied CNN to small text regions and combined various types of embedding to improve accuracy and were able to surpass the baseline, achieving a result with an error rate of 7.67% [Johnson and Zhang, 2014].

Meanwhile, Kalchbrenner, Grefenstette, and Blunsom [Kalchbrenner et al., 2014] adopted an easy applicable model for sentiment modelling of sentences by using a global pooling operation over linear sequence which enables the networks to handle sentences of varying length and different languages.

Learning word vector embedding is also an essential step in text learning models. Continuous vector representations of words from large datasets can be computed using neural networks at low computational cost [Mikolov et al., 2013]. However, most text sequences used in test are very different from the ones used for training, and as a result the vector representations obtained may not be as much useful as needed during test. Hence, Bengio, Ducharme, Vincent and Jauvin proposed the method of sending semantically neighboring word vectors of the training sentence back to the model [Bengio et al., 2003].

3 Dataset

Among all datasets examined by Kim, we choose to use "Rotten Tomatoes movie review" which was introduced by Bo Pang and Lillian Lee [Pang and Lee, 2005] to test the performance of our models. It contains 5331 movie reviews labeled as positive and 5331 reviews labeled as negative. Each data point consists of only one sentence. The light scale of the dataset makes it preferable such that we can test models without focusing on the linguistic aspect of the data.

3.1 Data Preprocessing

In order to tokenize the text input, we first clean up the sentences by replacing punctuation and contractions with a blank space followed by the punctuation or the contraction itself. Then, each sentence is split by blank spaces to form a list of words. A dictionary containing all words is created to count the total number of distinct words in the dataset. In total, we collect 18765 words as vocabulary.

The entire data set is shuffled and split to training set and validation set, with a ratio of 9:1, which is identical to the splitting ratio mentioned in Kim's work.

For the NBSVM model we implemented, N-grams are used to extract features, instead of dealing with only single words. Phrases work better than words in text classification because the order of words is more informative than single words [Wang et al., 2007], and N-grams are advantageous for detecting combination of words. For the purpose of this experiment, we explored uni-gram, bi-gram and tri-gram, as larger values of N make the features less representative of the information they contain.

4 Proposed Approach

We explored two of the baselines mentioned in the Kim's paper. All deep learning models are run using virtual TPU provided by Google Colaboratory (Colab), and non-deep learning model is run using Colab as well.

4.1 CNN

The first baseline model we implemented is CNN-rand. In this model, each word of each sentence in the dataset is represented by a vector, and 1D filters are applied to map several vectors to a scalar. As a result, each sentence maps to a vector which is then max-pooled and mapped to the targeted output. Figure 1 shows an example of the model. Since filters with different sizes are used to extract different features, several convolutions exist in the same layer, and the outputs of the convolutional blocks are concatenated and then flattened to act as a single output. Unlike other CNN models proposed

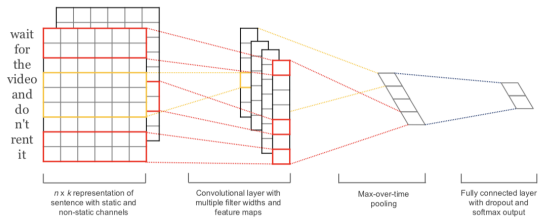

Figure 1: Model architecture with two channels for an example sentence. [Kim, 2014]

by Kim, the random baseline model does not use pre-trained word vectors for word embedding. Instead, the embedding layer randomly initializes vectors and modifies them during the training process. [Kim, 2014]

4.1.1 CNN-rand

Figure 4 shows the model architecture of CNN-rand implemented by us. For this model, we use the same hyperparameters as Kim's work. Three different filter sizes are used: 3, 4, 5, with each filter outputs 100 channels. The dropout rate is set to 0.5 as default, and the mini-batch size is set to 50. Since the longest sentence in the dataset has a size of 56, we use 50 as the embedding size.

All training process is done through Adam optimizer with hyperparameter tuning. [Kingma and Ba, 2014] The activation function for convolutional layer and fully connected layer is Rectified Linear Units (ReLU) to avoid negative outputs of hidden layers, and the activation function for output layer is Sigmoid so that the final output is in range [0,1]. Since the prediction should be either positive or negative, loss is computed using binary cross-entropy. To regularize the training, dropout layers are used in the models, in order to further reduce overfitting.

4.1.2 Modified CNN-rand

After reproducing the baseline model of Kim's work, we modified the original model because it overfits the training set. The new model as shown in Figure 5 uses 10 filters for each filter size, and the dropout rate is increased to 0.8, instead of 0.5, for the dropout layer after convolution.

4.2 NBSVM

Another baseline we built is Naive Bayes Support Vector Machine (NBSVM), which uses the element-wise products of log ratio and features as the input of SVM [Wang and Manning, 2012]. To compute the log ratio r, first we define V to be the set of all features, that is, the set of all words in the vocabulary list. Then define $f_j^{(i)}$ to be the number of times that word V_j occurs in training sentence i. Next we calculate two count vectors p and q such that

$$\mathbf{p} = \alpha + \sum_{i:y^{(i)}=1} \mathbf{f}^{(i)}$$
$$\mathbf{q} = \alpha + \sum_{i:y^{(i)}=-1} \mathbf{f}^{(i)}$$

Thus, we can evaluate the log ratio r as follows:

$$\mathbf{r} = \log\left(\frac{\mathbf{p}/||\mathbf{p}||_1}{\mathbf{q}/||\mathbf{q}||_1}\right)$$

With this ratio r, we rewrite the input of SVM by

$$\mathbf{x}^{(k)} = \tilde{\mathbf{f}}^{(k)}, \text{ where } \tilde{\mathbf{f}}^{(k)} = \hat{\mathbf{r}} \circ \hat{\mathbf{f}}^{(k)}$$

The model uses an interpolation which can be considered as a method of regularization. Rewrite the normal of the boundary w as the formula below, where $\bar{w} = \|w\|/|V|$.

$$\mathbf{w}' = (1-\beta)\bar{w} + \beta\mathbf{w}$$

This regularization procedure makes a choice between using the result from running Multinominal Naive Bayes and running SVM. The principle is to choose the result from Multinominal Naive Bayes under most circumstances unless SVM generates a confident result. We implement this model based on the work of Wang and Manning(2012). [Wang and Manning, 2012]

5 Results

5.1 CNN

Figure 2 shows the training loss and validation loss over epochs. From the plot, we discover that the original model is overfitting, which can be explained by that the model is too complex for this task. Hence we reduce the number of filters used to 10 for each convolution, and we increase the dropout rate of the second dropout layer to 0.8. The resulted learning curve is shown by Figure 3. We can

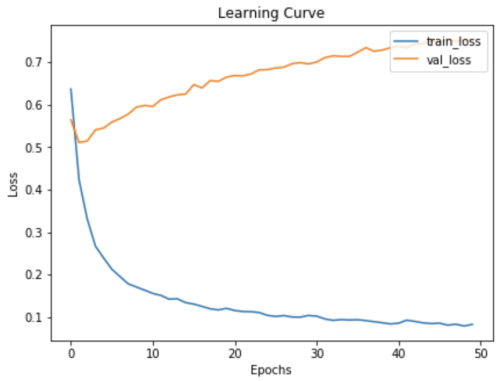

Figure 2: Learning curve over epochs of Original CNN-rand

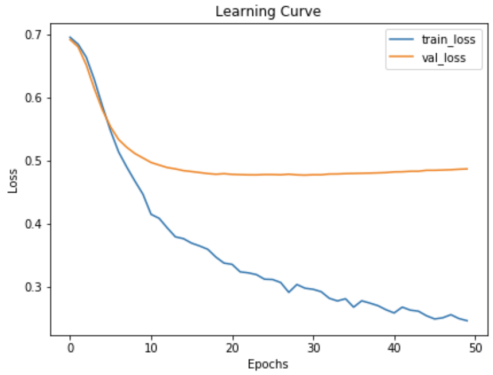

Figure 3: Learning curve over epochs of Modified CNN-rand

tell that the weights on validation set converge, and the overfitting is resolved.

	train_acc	**valid_acc**
CNN-rand	0.9682	0.7741
Modified CNN-rand	0.9028	0.7929

Table 1: Comparison between accuracy of original CNN-rand and modified CNN-rand

Table 1 shows the accuracy of two implemented CNN models. The accuracy mentioned in Kim's work of CNN-rand is 0.761 for the same dataset, with very little tuning [Kim, 2014]. After fine tuning, we achieve accuracy of 0.774 on validation set by using learning rate=0.01 and decay rate=0.055. The modified model outperforms the original one.

It achieves accuracy of 0.793 with the same learning rate and decay rate.

5.2 NBSVM

Train Accuracy	Validation Accuracy
1.0	0.8079

Table 2: Accuracy of NBSVM

Table 2 shows the result of NBSVM after hyperparameter tuning. The hyperparameter values used are tolerance=1 and C=1 for the linear SVC. Penalty is calculated by L2 loss function. Using both uni-gram and bi-gram results in better performance, compared to using uni-gram or bi-gram alone, or using uni-gram, bi-gram and tri-gram together.

5.3 Time Complexity

Table 3 presents the running time of three models. Deep learning models, CNN-rand and modified CNN-rand, require more time to converge to optimal weights, while the simpler algorithm, NBSVM, takes much less time to compute. Between the two CNN models, our modified model has better efficiency, since the model architecture is less complex. Both CNN models run 50 epochs.

	Runtime (s)
CNN-rand	1073.7
Modified CNN-rand	598.6
NBSVM	0.0906

Table 3: Comparison of runtime between models

6 Discussion and Conclusion

Overall, the performance of NBSVM is better than that of the CNN baseline models, and the computation time is much shorter. Between the two tested CNN models, our modified model is less complex, yet has higher accuracy and efficiency. It indicates that simple algorithms with less computation complexity may outperform more complicated algorithms for simple text classification like the Rotten Tomatoes task. Even though the learning curve of modified CNN-rand does not indicate overfitting of the model, the validation loss stops decreasing when the training loss continues to drop, which implies that the model is still too complex for the given task. As mentioned in Kim's work, the proposed CNN models outperform other algorithms mainly due to the use of pre-trained word vectors [Kim, 2014]. Hence, the CNN architecture itself does not play a major role in elevating performances. It is the word embedding techniques that should be paid more attention to in further studies of applying CNN to text classification problems.

For possible further improvement of the CNN-rand model, NLP techniques such as lemmatization and removal of stop words can be applied during data preprocessing phase so that the collected dictionary is cleaner and shows more semantic information because different forms or tenses of words are unified to their lemmas.

7 Statement of Contribution

The following states each person's contribution to this work:

- Zirui Kuai: Implementation and tuning

of NBSVM, tuning of CNN-rand and modified CNN-rand;

- Kaylee Zhao: Implementation of CNN-rand and modified CNN-rand;
- Yihe Zhang: Implementation of CNN-rand and modified CNN-rand;

All members contributed to the write-up.

References

[Bengio et al., 2003] Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

[Collobert et al., 2011] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537.

[Graves et al., 2013] Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE.

[Johnson and Zhang, 2014] Johnson, R. and Zhang, T. (2014). Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*.

[Kalchbrenner et al., 2014] Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

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

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Pang and Lee, 2005] Pang, B. and Lee, L. (2005). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 115–124. Association for Computational Linguistics.

[Shen et al., 2014] Shen, Y., He, X., Gao, J., Deng, L., and Mesnil, G. (2014). Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 373–374. ACM.

[Wang and Manning, 2012] Wang, S. and Manning, C. D. (2012). Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th annual meeting of the association for computational linguistics: Short papers-volume 2*, pages 90–94. Association for Computational Linguistics.

[Wang et al., 2007] Wang, X., McCallum, A., and Wei, X. (2007). Topical n-grams:

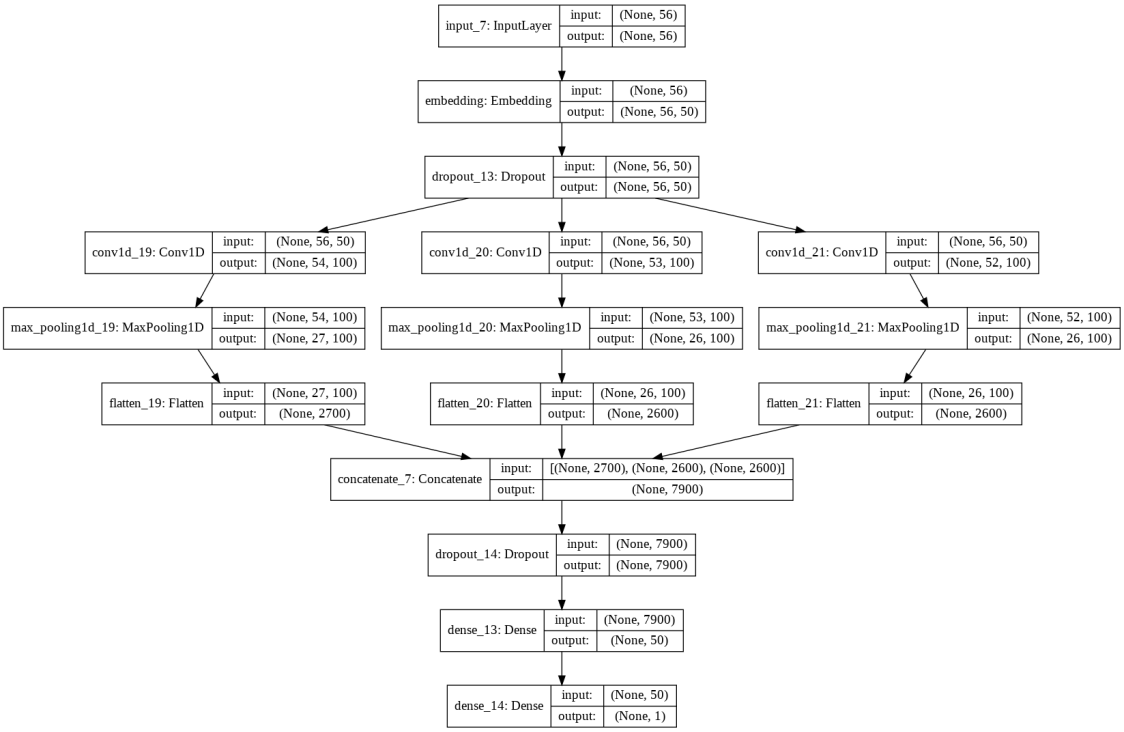

Figure 4: Model architecture of Reproduced CNN-rand

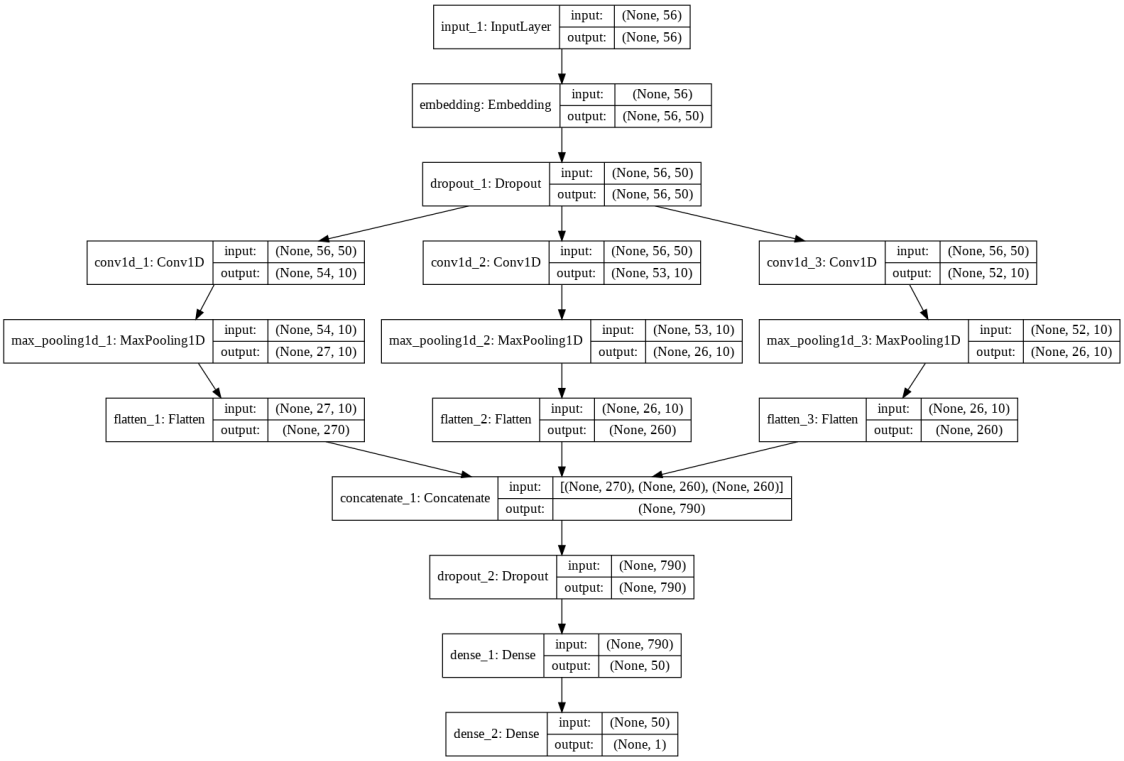

Figure 5: Model architecture of modified CNN-rand

Phrase and topic discovery, with an application to information retrieval. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 697–702. IEEE.

[Yih et al., 2014] Yih, W.-t., He, X., and Meek, C. (2014). Semantic parsing for single-relation question answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 643–648.