Analysis of Convolutional Neural Networks for Sentence Classification

Yuxiang Ma, Mathieu Rundstrom, Shenshun Yao

McGill University, Montreal, QC, CAN

Abstract

In this paper, we investigate the performance of convolutional neural networks (CNN) developed by Yoon Kim in his *Convolutional Neural Networks for Sentence Classification*[3]. We reproduced the work of Yoon Kim while keeping the same baselines setting. We found that our reproduced accuracy outperformed his results on SST-1, but under-performed on SST-2 and TREC, due to the differences in aspects of both deep learning frameworks and implementation of fundamental optimizers. To investigate the effect of optimizers on training process and result, we changed the optimizer of the CNN models from Adadelta to Adam, which resulted in significantly better performance.

1. Introduction

We reproduced a CNN model for sentence classification by Yoon Kim[3] and purposed a new baseline which improves the performance of CNN models.

We started by introducing several variants of CNN model as well as some related work, for example the recently best results. Then we investigated how Yoon Kim constructed the CNN model by transforming the words into vectors, using convolutional layer with multiple filter widths and feature map, and max-over-time pooling [3]. Models were tested on SST1, SST2, TREC, for these datasets have already been officially separated into training, validation and test sets. We found that our results only beat Yoon Kim's on SST1. Therefore, we continued to investigate how to further improve the performance on SST1 set by modifying the baselines of these CNN models. Our experiments illustrated training through stochastic gradient descent with Adam update[4] can achieve a better performance.

2. Related work

Sentiment classification is a fundamental problem in machine learning. CNNs have gained a lot of attention the past few years, especially in the area of image classification. In his paper, Kim shows that even a simple CNN achieves good result on sentence classification. Below is an image of the general model architecture, which was based on a model from 2011 by Collobert et al. [2]. It represents the CNN-multichannel variant, but is similar enough to the other variants, except for the fact that it has two channels instead of one.

Different variants of the CNN are introduced in the paper, with main differences being the initialization of hyper parameters. The four variants are as follows

1. **CNN-rand**, the baseline model, where all words were randomly initialized and then modified during training.

Email addresses: yuxiang.ma@mail.mcgill.ca (Yuxiang Ma), mathieu.runstrom@mail.mcgill.ca (Mathieu Rundstrom), shenshun.yao@mcgill.ca (Shenshun Yao)

[1]This is the report for miniproject 4 of Comp 551 Applied Machine learning, written by group 8.

[2]Yuxiang Mainly contributed to model construction, Yuxiang and Shenshun ran the experiments and tested the results. The report was written together.

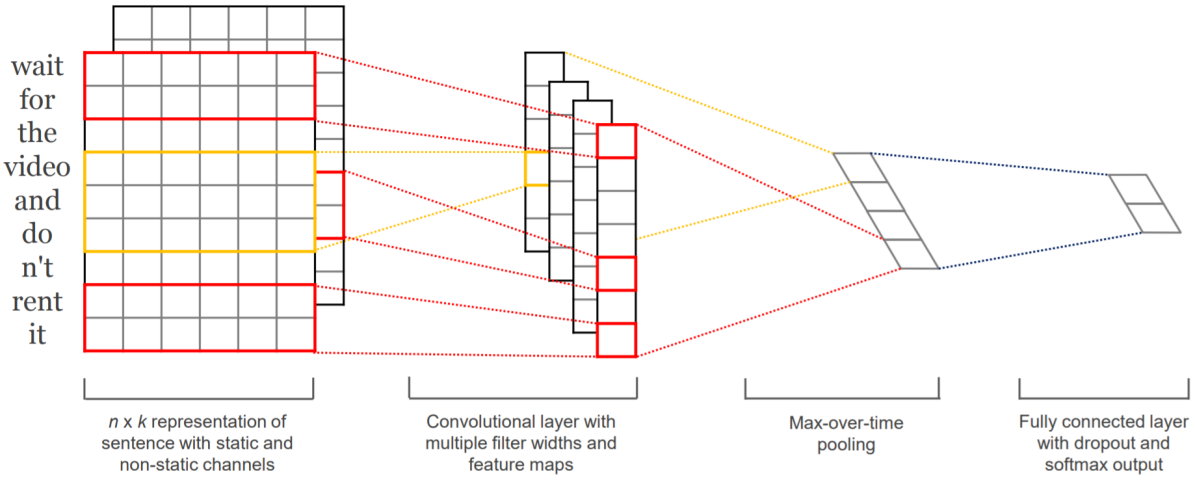

Figure 1: Example model, taken from Kim's paper [3].

2. **CNN-static**, where all words were initialized using word2vec[3], and the other model parameters were learned through training.

3. **CNN-non-static** the same as above; however, fine tuning for each task.

4. **CNN-multichannel**, which uses two sets of words, where filters are applied to both but backpropagation is ran on one. Also initialized using word2vec.

At the time of publishing (fall 2014), these model variants all achieved state of the art or close to state of the art results on the seven datasets mentioned in Kim's paper. We will investigate performance on three of these datasets: SST-1, SST-2, and TREC. Specifically, the best performing model on SST-1 was the static variant, which had an accuracy of 48.0, compared to the 48.7 of Le and Mikolov's Paragraph-Vec [5]. On the SST-2 dataset, the CNN-multichannel variant had the best known accuracy: 88.1. However, we remark that more recently, in 2018 and early 2019, great advances have been made using deep learning models. Notably, Peters et. al. and Liu et al. have achieved a accuracies of 54.7 and 95.6 respectively on SST-1 and SST-2 [7] [8]. Moreover, researchers from Google have achieved an accuracy of 98.07 on TREC [1]. At the time of writing this paper, these are the best results known to us.

3. Dataset and setup

Two of the datasets we use are based on the Stanford Sentiment Treebank (SST) [9]. The data in SST is taken from online movie reviews. It includes 11,855 sentences, containing 215,154 unique phrases. SST has already made train and test splits. There are two variants of SST, namely SST-1, which has five labels: negative, somewhat negative, neutral, positive or somewhat positive. The SST-2 data set is the same as the SST-1, except the neutral data is removed and negative, somewhat negative are clumped together as one label, similarly with positive and somewhat positive. Since part of the data is removed for SST-2, it contains 9,613 sentences. It's worth to remark was that these sentiments were obtained by human labeling. The last dataset we base our analysis on is TREC, a dataset for question classification consisting of 5952 fact-based questions divided into six broad semantic categories, the categories being: abbreviation, description, entity, human, location, numeric [6]. For example, under the entity category, one might find

[3]A publicly available list of word vectors trained on 100 billion words taken from Google News

questions such as " *What type of currency is used in Australia ?*", or under human, one might find questions such as " *Who was the first governor of Alaska ?*".

4. Proposed approach

In this section, we further describe the baseline model from Kim's paper (seen in Figure 1 is the model, but with two channels) in more detail and with more rigour. Firstly, a word vector is a way to represent a word through a projection onto some denser space such that one can make sense of a distance between these vectors; words that are semantically close should be a small, say euclidean, distance apart in this representation. We define, as in the paper we are comparing to, $x_i \in \mathbb{R}^k$ the i-th word of a sentence. Let $\oplus$ denote the concatenation operator. We introduce notation for concatenating words: $\mathbf{x}_{i:i+j} = x_i \oplus x_{i+1} \oplus \cdots \oplus x_{i+j}$. Naturally, a sentence of length n is denoted by $\mathbf{x}_{1:n} := x_1 \oplus x_2 \oplus \cdots \oplus x_n$. What we have described is similar to the leftmost block in Figure 1.

The second block in the model, the convolution layer, is comprised of two steps. First, a filter $\mathbf{w} \in \mathbb{R}^{hk}$ is applied to a windows of h concatenated words i.e. $j = h - 1$ in the above form, onto which a bias term b is added. Second, a non-linear function f is applied to this; the total described by the following

$$c_i := f(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b).$$

This is done over all possible windows of size h, leading to a feature map $c := [c_1, \ldots, c_{n-h+1}]$.

Third is a max-over-time pooling operation, which is just a max pooling operation with "time" meaning the position in the sentence. In contrast with CNNs used in image processing/classification, where the input images are of the same size, we here have sentences of different lengths. To be able to apply one function to all of these sentences, it is reasonable to convert the result from the previous layer into a single number; the maximum over the whole feature map. Mathematically, we denote this by $\hat{c} := \max \mathbf{c}$. We note that the model uses multiple filters with different values for h (3, 4, 5) to obtain multiple features, forming the penultimate layer, which feeds into a fully connected softmax layer whose output is the probability distribution over labels. Finally, dropout with $p = 0.5$ is used during training (disabled during testing), and the l_2-norms of the weight vectors are set to 3 whenever $||\mathbf{w}||_2 > 3$ after a gradient descent step. Kim obtained these values via grid search on the SST-2 dataset. Grid search is a method of attempting to obtain optimal hyperparameters through training different models with many different combinations of hyperparameters over a matrix (grid) of hyperparameters, and choosing the combination which performs best on a validation set (or through some other type of performance evaluation of the model). Mini-batches of size 50 are used in training, together with stochastic gradient descent and the adaptive[4] Adadelta update rule, introduced by Zeiler.

As the figures below (Figure 2-5) show, gradient flows are different among all proposed model variants. In CNN-rand, embedding $\mathbf{w}$ was initialized with a random vector and optimized with Adadelta, and error was back propagated as part of computation graph. In comparison, CNN-static has a non-trainable embedding $\mathbf{w}$, for there is no optimizer shown, and no error back propagated to $\mathbf{w}$ in the graph. As for the non-static model and the multi-channel model, the gradient $\mathbf{w}$ was initialized with pre-trained vectors (using `word2vec`, as described previously). Additionally, the non-static channel of CNN-multichannel and the input channel of CNN-non-static were optimized with Adadelta, with error back propagated to their $\mathbf{w}$s, while $\mathbf{w}_{\text{static}}$ channel of Multi-channel Model has no optimizer and, hence, error was not back propagated to $\mathbf{w}_{\text{static}}$.

To implement l_2 norm constrains that $\mathbf{w}$ should not be larger than 3 during training, $\mathbf{w}$ was computed and re-scaled to norm less than 3 (if it is larger than 3) before applied to next iteration using clipping by norm of Tensorflow [5].

[4] adaptive refers to an adaptive learning rate, which changes depending on how the training progresses

[5] https://www.tensorflow.org/api_docs/python/tf/clip_by_norm)

As for evaluation, and similarly to Kim, we altered dropout keep probability to 1.0 and re scaled all **w**s in network with coefficient equal to previous dropout keep probability p, such that $\mathbf{w}_{\text{eval}} = p\mathbf{w}_{\text{train}}$ to enable whole network during evaluation while keeping the effect of tuning dropout rate during training [10].

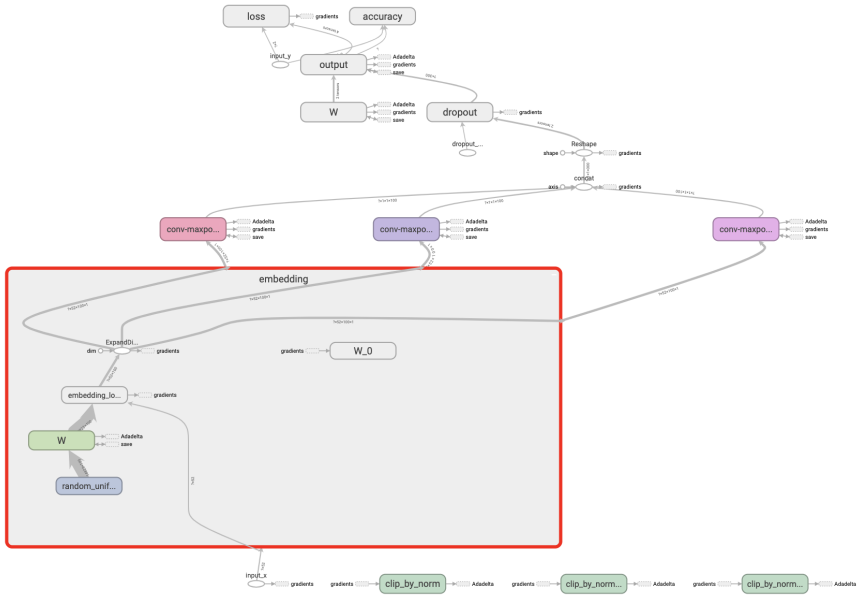

Figure 2: Computation Graph of Random Model

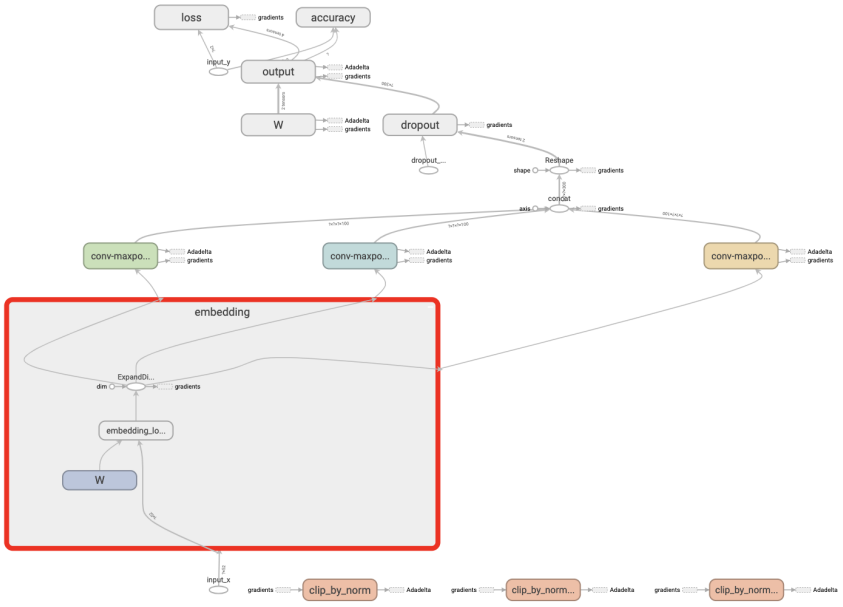

Figure 3: Computation Graph of Static Model

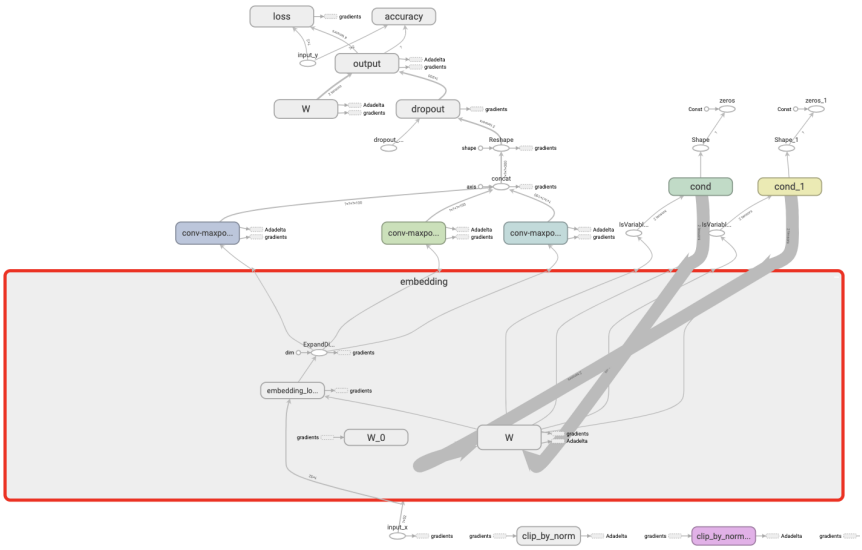

Figure 4: Computation Graph of Non-static Model

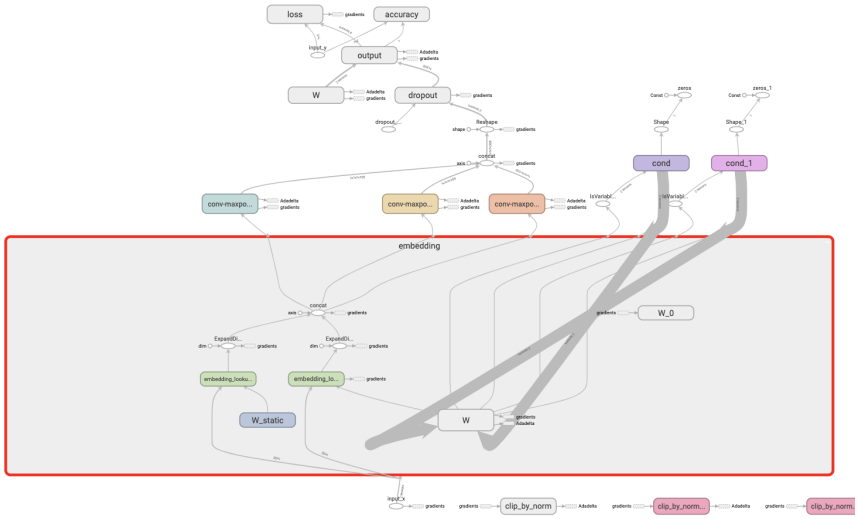

Figure 5: Computation Graph of Multi-Channel Model

5. Results

In this part, we run several experiments to test the performance of our reproduced models, and we also propose a new baseline that achieves stronger performance.

*5.1. **Reproduction of Yoon Kim's results***

We started by reproducing a subset of Yoon Kim's results[3] on SST-1, SST-2 and TREC.

Model	SST-1	SST-2	TREC
CNN-rand	47.06	80.72	87.20
CNN-static	49.41	77.27	90.60
CNN-non-static	51.49	79.52	91.60
CNN-multichannel	50.31	80.06	90.60

Table 1: Results of our reproduced CNN models

As stated in Table 1, we observed that on the one hand, our accuracies for the four different models on the SST-1 dataset beat Yoon Kim's results[3]. On the other hand, our results could not reach the accuracies which Yoon Kim mentioned in his paper [3] on SST-2 and TREC. The differences between our results and Yoon Kim's results could be caused by the implementation of optimizer. Yoon Kim wrote the Adadelta optimizer by himself (refer to his Github[6]), whereas we implemented it using the package provided by Tensorflow[7]. Therefore, we only report the performance on SST-1 in the following experiments, to check whether the performance is improved or not once we modify the baseline.

5.2. ***Training optimizer: Adadelta vs. Adam***

In this altered version of the model, we use the Adam algorithm as our optimization algorithm, which is already implemented in Tensorflow[8], and taken from the works of Kingma and Ba [4]. It is an alrogithm "for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments", which, according to them, outperforms other stochastic optimization models and has become the standard adaptive optimization algorithm. Since we suspected that the choice of optimizer might affect the performance of the CNN models, this implemantation change seemed justified. Yoon Kim[3] selected Adadelta as his optimizer, while we kept the same hyper-parameter setting with Yoon Kim's model[3] but only changed the optimizer to Adam. Therefore, our results from Table 2 illustrates that Adam is a better optimizer than Adadelta in this case.

Model	SST-1
CNN-rand	51.76
CNN-static	51.99
CNN-non-static	51.67
CNN-multichannel	52.08

Table 2: Results of CNN models using Adam optimizer on SST-1

As shown below, the Adam Optimizer converged faster, plateauing in about half the time of that of its counterpart, and with smaller fluctuation than the Adadelta Optimizer on SST-2 random training process. This is apparent in Figure 6, as the training accuracy grows faster in the Adam optimizer case than in the Adadelta case, and Adam has a lower training accuracy variance than Adadelta.

[6]https://github.com/yoonkim/CNN_sentence/blob/master/conv_net_sentence.py
[7]https://www.tensorflow.org/api_docs/python/tf/train/AdadeltaOptimizer
[8]https://www.tensorflow.org/api_docs/python/tf/train/AdadeltaOptimizer

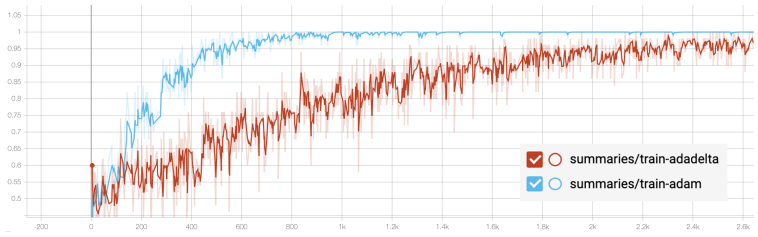

Figure 6: Convergence of Adam and Adadelta Optimizer on SST2 Random Training Process

6. Discussion and Conclusion

In this project, we developed a model for classifying text in SST1, SST2, and TREC datasets using Convolutional Neural Networks. The architecture consists of an input layer of 1 or 2 channels input with width of embedding dimension and height of number of words in the document, set of parallel convolution layer and max polling layer with various lengths, and a fully connected layer with dropout rate to prevent overfitting, and an output layer with cross entropy loss. The Adam optimizer was used and showed significant improvement model performance, compared to the Adadelta optimizer in original paper. According to result on the test set, we concluded that CNN with 3, 4, 5 heights windows, gradient clipping with maximum 3.0 $\|\mathbf{w}\|_2$ norm, 300 embedding size, and Adam optimizer is the optimal model of this type.

First, the Adam optimizer is a better optimizer than Adadelta under Tensorflow for Sentence Classification. Adam converges faster than Adadelta Optimizer on the validation set and gives higher output accuracy in test set, as seen in the above figure. In general, the advantage of using adaptive learning is that we do not need to focus on subtle tuning of the learning rate but rather focus on the model architecture and the algorithm.

Second, different implementation of deep learning framework might affect performance. There is still a difference in performance between our Tensorflow implementation and Kim's Theano implementation even using same architecture. We also had different performance on 3 datasets even with the same hyper parameters. The difference might due to implementation of SGD with Adadelta update in Tensorflow and Kim's own model, but further investigation is required to conclude such a thing. Therefore, further re-implementation should consider both the platform and the model.

According to Yoon Kim's Github, Ye Zhang[11] has written a paper doing an extensive analysis of model variants (e.g. filter widths, k-max pooling, word2vec vs Glove, etc.) and their effect on performance. Inspired by Ye Zhang's paper[11], we then run two more experiments to see the effect of modified some simple baselines. Due to limitation in time however, we have not implemented such a intensive hyper-parameter tuning. Nevertheless, this seems like a logical next step in further researching CNNs in sentence classification.

In the future, we could investigate using existing computer vision networks, such as DenseNet and SqeezeNet under some fine tuning. Also, the stability, performance and run time of simpler, more traditional machine learning methods, for example, NBSVM and MNB could also be of interest.

References

[1] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. Universal sentence encoder for english. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[2] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November 2011.

[3] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751, 2014.

[4] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[5] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1188–II–1196. JMLR.org, 2014.

[6] Xin Li and Dan Roth. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[7] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. *CoRR*, abs/1901.11504, 2019.

[8] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[9] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[10] Matthew D. Zeiler. Adadelta: An adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.

[11] Yingjie Zhang and Byron C. Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. In *IJCNLP*, 2017.