TAF Journal of Advances in Technology and Engineering Studies JATER
2016, **2**(5): 156-163

PRIMARY RESEARCH

Batch size for training convolutional neural networks for sentence classification

Nabeel Zuhair Tawfeeq Abdulnabi [1*], **Oğuz Altun** [2]

[1, 2] Computer Engineering Department, Yildiz Technical University, Istanbul, Turkey

Index Terms
Convolutional Neural Network
Sentence Classification
Word embedding

Received: 3 August 2016
Accepted: 10 September 2016
Published: 27 October 2016

***Abstract*—** Sentence classification of shortened text such as single sentences of movie review is a hard subject because of the limited finite information that they normally contain. We present a Convolutional Neural Network (CNN) architecture and better hyper-parameter values for learning sentence classification with no preprocessing on small sized data. The CNN used in this work have multiple stages. First the input layer consist of sentence concatenated word embedding. Then followed by convolutional layer with different filter sizes for learning sentence level features, followed by max-pooling layer which concatenate features to form final feature vector. Lastly a softmax classifier is used. In our work we allow network to handle arbitrarily batch size with different dropout ratios, which is gave us an excellent way to regularize our CNN and block neurons from co-adapting and impose them to learn useful features. By using CNN with multi filter sizes we can detect specific features such as existence of negations like "not amazing". Our approach achieves state-of-the-art result for sentence sentiment prediction in both binary positive/negative classification.

I. INTRODUCTION

In Natural Language Processing (NLP), most of the work with deep learning are deal with learning word vector embedding by using a neural network [1].

However, CNN which is a neural network that shares their parameter across space, considered to be responsible for a major breakthrough in sentence classification. Recently researchers started to employ CNN in NLP and they got promising results, especially in sentence classification [2].

However, in order to get a better understanding of CNN we have to think of it as a sliding window deployed to a matrix. And the shared by the computation units in the same layer. This weight shared enables learning valuable features regardless of their location, while preserving of their location where do beneficial features appear.

However, CNN for sentence classification considers quite powerful because it learns the way to weight individual words in a fixed size in order to produce useful features for a specific task.

II. RELATED WORK

Recently, [3] presented away to train simple convolutional neural networks with one layer of the convolution on top of word vectors obtained from unsupervised neural language model, to save word vectors static and try to learning the other hyperparameters of the model. However, they found that features extracted obtained from a pre-trained deep learning model get a good result in the different task. [4] introduced an excellent study on character-level ConvNet for text classification. Also, make comparisons between ConvNet and against traditional methods like a bag of the words and n-gram. However, his result illustrate that character-level of convolutional neural networks is an efficient method. On the other hand [5] reported a good way to model short texts using semantic clustering and CNN. They find that model uses pre-trained word embedding will introduce extra knowledge, and multi-scale SUs in short texts are detected. [6] introduced a model to capture both semantic and sentiment similarities among words, The semantic component of their model learns word vectors via an-

[*] Corresponding author: Nabeel Zuhair Tawfeeq Abdulnabi
[†] Email: nabil78.nz@gmail.com

unsupervised probabilistic model of documents, they made an extended the unsupervised model to incorporate sentiment information and showed how this extended model can leverage the abundance of sentiment-labeled texts available online to yield word representations that capture both sentiment and semantic relations.

However, the reported method performed better than LDA, which models latent topics directly [7] they impact of deploying machine learning mechanism to the sentence classification problem. The face challenge in detected review that not contain keyword to express sentiment polarity like ("how could anyone sit through this movie ") because it is not include any word to give meaning for negative. They find out that standard machine learning techniques definitively outperform human-produced baselines. [8] reported a method that can utilize the word order out sentence, which apply convolutional neural network. However, use of one dimension structure (word order) of the document in which each part of convolution layer deal with a specific part or region of document (sequence of words). [9] applied bow-CNN and seq-CNN both methods shows outperform comparing with baseline approach on all datasets they used also they find that seq-CNN get better performance than bow-CNN in sentiment analysis but on the other hand bow-CNN outperform seq-CNN in topic classification. Notably, in most of the convolution neural network on text classification, the input layer is the vectors of the transformed word of the sentence which either trained by CNN or another approach like (word2vec) [10].

Our work is close to the work [11] which examine convolutional neural network on top of pre-trained word vectors whereas our work start training convolutional neural network and learning to embed from scratch and we use one input channel.

III. OUR MODEL

The convolutional neural network we built looks as figure 1, the first layer is the input layer which embeds the words into low dimension vector. Followed by a convolutional layer which is performed convolutions over the embedding word vectors by utilize multiple filter sizes or sometimes called kernel or sliding window. For instance sliding over 3, 4 or 5 words at a time. The next layer is max-pool layer which is responsible of max-pool the result or stack the result from convolutional layer into a long feature vector, later on we add dropout for regularization, and finally the softmax layer that can make classify the result into binary classification.

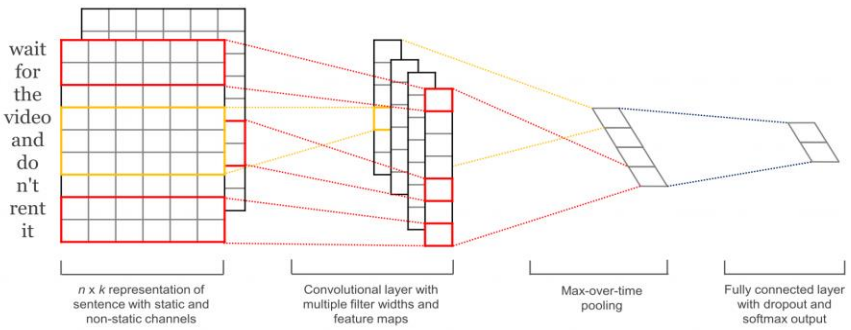

Fig. 1 . Convolutional neural network architecture [13]

Convolution layer will interact with the output of neurons that are connected to local space in the input, each computing a dot product between their weights and a small region they are connected to in the input volume.

The activation function is a nonlinear activation like Relu applied to the outcome of convolution layer. In common feed forward neural network, we connect each input neurone with output one in the followed layer (fully connected layer. However in convolutional neural network actually we convolutions over the input layer to get the output. Through training, a convolutional neural network automatically learns the filters value depend on the job we want to perform.

Filters that slide over full rows of the matrix (words). Thus, the "width" of our filters is generally the same as the width of the input matrix. The height, or region size, may

ISSN: 2414-4592
DOI: 10.20474/jater-2.5.3

vary, but sliding windows over 2-5 words at a time is looks good figure 2 illustrate the network.

In figure 2, each kernel applies convolution on sentence matrix and produce a variant length of feature maps. Followed by pooling to apply on each map, and the max value from each feature map is captured feature are concatenated to feature vector and then followed by softmax layer which deal with these feature as input and utilize it to classify the sentence. In our work we consider binary classification so there is to possibly zero for negative polarity and one for positive sentence.

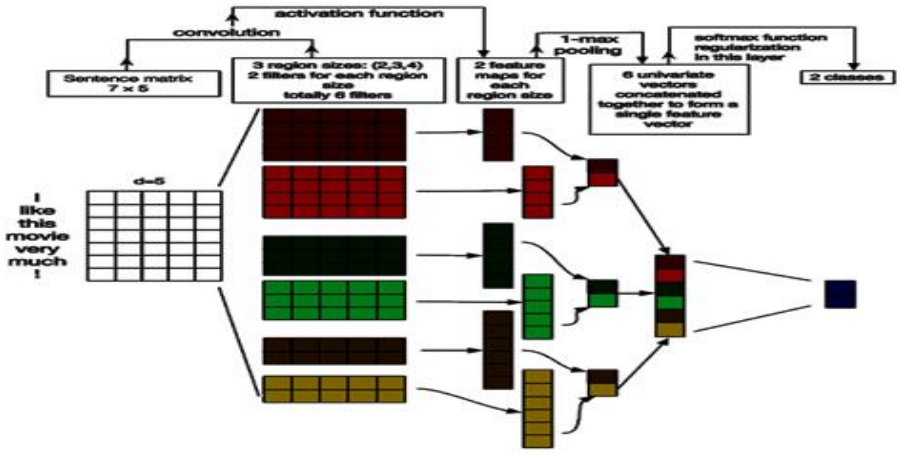

Fig. 2 . Illustration of a Convolutional Neural Network (CNN) architecture for sentence classification [4]

A. CNN Architecture

We start by tokenization step in which sentence is convert to a sentence matrix, the rows represent word vector of each token. We refer to the word vectors by D. If the length of a given sentence is S, then the dimensionality of the sentence matrix is S * D. after that we can deal with sentence matrix and then perform convolution on this matrix using different size of kernel or filter. However, here we have to use a kernel with size equal to the width of the dimensionality of the word vectors. But the high of the kernel can be vary so we can refer to the high of the kernel as region size of the kernel. In traditional feed forward neural network if we have 4 input node and 3 feature space then we have 12 base or parameters, however in CNN if we have 4 input node and the length of the kernel is 2, in this case we have 6 parameters (3*2).

Therefore, in normal neural network every output unit interacts with every input unit. However in convolutional neural networks typically have sparse connectivity (sparse weights), this is done by making the kernel smaller than input. However, in CNN we share the parameters this reduce the complexity of the network. Assume that there is a kernel parameterized by the weight matrix W with region size R. So W will contain R*D parameters to be estimated. We suppose that a matrix of sentenced as $M \in RSt^{\text{t×D}}$ and use M[i : j] to represent the sub-matrix of M from row i to row j. The output sequence o ∈ RS−R+1 of the convolution operator is obtained by repeatedly applying the filter on sub-matrices of A:

$O_i = w[i : i + R-1]$

Where i = 1 . . . S − R + 1, and is the element wise product between the sub-matrix and the kernel (a sum over element-wise multiplications). And then we add a bias term b ∈ R and an activation function f to each o i, motivate the feature map c ∈ R S−R+1 for this kernel:

$C_i = f(o_i + b)$.

The dimensionality of the feature map generated by each kernel will be different as a function of the sentence length and the kernel region size. A pooling stage mean we are going to do summary statistic of our output for example take the maximum value of the result of nonlinear stage or detecting stage, so we can use pooling for down sampling and this lead to minimize the complexity of the network. Any classifier need a fixed size of input so by pooling stage we down sampling and make them fixed size because it's going to summarize statistically. A pooling function is thus applied to each feature map to produce a fixed-length vector.

ISSN: 2414-4592
DOI: 10.20474/jater-2.5.3

A traditional strategy is 1-max pooling [11] which extracts a scalar from each feature map. The outputs produced from each kernel map can be concatenated into a fixed-length, 'top-level' feature vector, and then the result fed to a softmax function to generate the final classification. At this softmax layer, one may apply 'dropout' [12]. As a means of regularization. This entails randomly setting values in the weight vector to 0. Our aim is to reduce cross-entropy loss. The parameters to be learned include the weight vector(s) of the kernel, the bias term in the activation function, and the weight vector of the softmax function.

IV. CONVOLUTIONAL NEURAL NETWORK HYPERPARAMETERS

In order understand how CNN are deal with natural language processing we need to know about hyperparameters.

B. Narrow VS Wide Convolution

May ask how would apply the kernel to the first element of a matrix that does not have any neighbour elements to the top and left? We can use zero-padding, so all elements which locate outside the matrix would be zero. Therefore, we can apply the kernel to each element of our input matrix. However, if we adding zero padding this will call wide convolution, and a non-zero padding will be called narrow convolution.

C. Stride Size

The other hyperparameter of our CNN is called stride size, which can be defined as how much we want to shift our kernel at each step. And a successive applied of kernel overlapped. Whenever the size of stride is large this mean less applied of kernel and also small output size. As illustrate in figure 4.

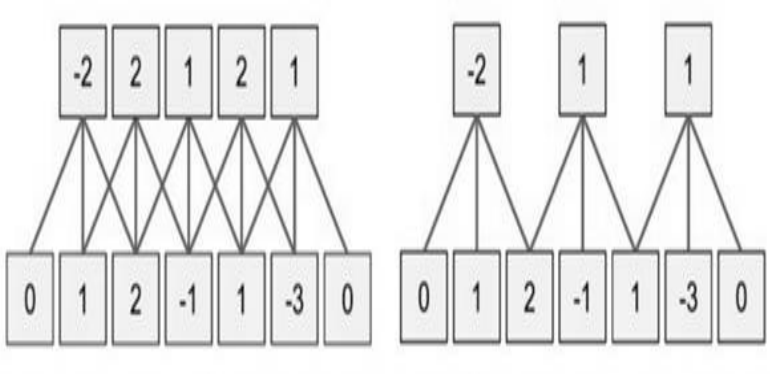

Fig. 3 . Convolution Stride Size. Left: Stride size 1. Right: Stride size 2 [14]

D. Pooling Layer

The pooling layer which considers the main key is applied after convolutional layer. Pooling layers is subsample their input. A traditional way to do pooling is by using a max function to the output of each kernel. In natural language processing, we use pooling over the all out, yielding a single number for each kernel. The advantage of using pooling layer is to produce a fixed size output matrix, which is wanted for classification. For instance, if we have 100-kernel and we apply max pooling to each, we will get 100-dimensional output, regardless the size of our kernel, or either the size of our input. This gives us permission to apply different size sentences. By applying pooling actually, we minimize the dimensionality of output but keep the useful features. We can figure out each kernel as detecting particular feature like capturing sentence has negative meaning like "not good". However, after applying max function we can keep information if the feature appears in the sentence or not.

V. TRAINING A TEXT EMBEDDING MODEL

Imagine that you want to classify a document we are going to have to look at the words in that document to figure out that. The words are really difficult there are a lots of them and most of them you never, ever see. In fact, the ones that you rarely see tend to be most important ones.

ISSN: 2414-4592
DOI: 10.20474/jater-2.5.3

For deep learning more events like that are a problem. We like to have lots of training examples to learn from. Another problem is that we often use different words to mean almost the same thing. for example " cat" or we can say ' kitty ", they are not the same , but they mean similar things , so when we have things that are similar , we really wants like to share parameter between them , if we want to share anything between kitty and cat , we are going to have to learn that they are related. Therefore, we would like to see those important words often enough, to be able to learn the meaning automatically. And would also like to learn how words relate to each other so that we can share parameters between them. And that would mean collecting a lot of label that there are many way to use the idea that say similar words occur in similar contexts. in our case we are going to map words to small vectors called embedding when are going to be closed to each other when words have similar meanings , and far apart when they don't . Embedding solves of the sparsity problem. once we have embedded our word into this small vector now we have a word representation where all the catlike thing like cats , kitties , kittens , pets, lions are all represented vectors that very similar.

A. Examine the influence of Dropout Training for Convolutional Neural Networks

Dropout is a modern regularization technique that has been more lately applied in deep learning. we define regularization as minimize free parameters of the network and keep the optimization. It is the way to avoid overfitting in the training stage. However, dropout work like this, imagine that we have one layer that connects to another layer the values that go from one layer to the next are often called activation function. Take that activation and randomly for every example, we train our network on we set half of them to zero. Completely randomly, we basically take half of the data that is flowing through our network and just destroy it. And then randomly again. So what happen with drop out? Our network can never rely on any given activation to be present because they might be squashed at any given moment. Therefore it is forced to learn redundant representation for everything to make sure that at least some of the information remains. One activation get smashed but there is always one or more that do the same job and that do not kill. Everything remains fine at the end. Forcing our network to learn redundant representation might sound very inefficient. But in practice, it makes thing more robust and prevents over-fitting. It also makes our network act as if taking the consensus over an ensemble of networks. Which is always a good way to improve performance Dropout is one of the most important techniques to emerge in the last few years.

B. Batch Size

Batch size consider one of the hyperparameters that can tuning during training our neural network. So how to set the optimal batch size? Let's take the two opposite side, on one side each gradient descent step is applying to the all dataset. We're computing the gradients for all example. In this case we know exactly the directly towards a local minimum. We don't waste time going the wrong direction. But in this way, the computation on all dataset will be very expensive. So let try the other side of our scenario, a batch size of just 1 example. In this case, the gradient of that example may take you entirely the wrong direction. However, the cost of computing the one gradient was very cheap. And averaging over a batch of 10, 100, 1000 example is going to generate a gradient that is a more sense. In our work in order to find the better batch size we itrate oure modle multiple time by using intition of the trail and error. We tried different batch size to find out the optimal batch size for convergence.

C. Dataset

The dataset we'll use in this paper is the movie Review data from Rotten Tomatoes. The dataset contains 10,662 example review sentences, half positive and half negative. The dataset has a vocabulary of size around 20k. Because of our data set is pretty small we're likely to overfit with a powerful model. And also the dataset does not split to train and test so we split the dataset as follow: 20% testing, 20% validation and 60% training. After that we deal with pre-processing on dataset which is:

• load positive and negative sentence from the raw data files.

• Clean the text data by converting the all the upper case letters to the lower case get a ride from the weight space and so on.

• Pad each sentence to the maximum sentence length, which turns out to be 59. We append special <PAD> tokens to all other sentences to make them 59 words. Padding sentences to the same length is powerful since it allows us to efficiently batch our data since each example in a batch must be of the same length.

ISSN: 2414-4592
DOI: 10.20474/jater-2.5.3

• Build a vocabulary index and map each word to an integer between 0 and 18,765 (the vocabulary size). Each sentence becomes a vector of integers.

VII. EXPERIMENT

Experiment for the implementation part and result test, we used python language based on python compiler under Linux-os.we have to take care about performance evaluation of our network in term of accuracy and regularization by tuning hyperparameter batch size and dropout. The objective of our work is to classify the sentence of movie review into binary classification. So we have two output from our network 1 for positive sentence and zero for negative sentence. We examine our model with different hyperparameters (batch size and dropout) and we try tuning these parameters in order to find the better convergence for our model. First we train our model using batch size 64 and dropout equal to 0.5 to evaluate the net with respect to the accuracy. The traditional way to regularize CNN is L2 and dropout in our work dropout. We examine our network with a variable value between 0.1 and 0.5, and the measure the accuracy for each value of dropout. We also use different kernel size (filter) (3, 4 and 5). We test the network with different values of hyperparameters in order to find optimal values for these parameters. However, we defining loss function to compute the error of our model, and our aim is to minimize it. Therefore we apply cross_entropy loss which measures the loss for each class, given the true label sample and our output scores of the network. After that, we take the average of the losses. In table 1 we examine our model in different batch size (8, 16, 32, and 64). We get higher accuracy in batch size =32. In table 2 we evaluate the influence of batch size to the loss rate in the network. However, we get the minimum loss when batch size was 64.

TABLE 1
IMPACT OF BATCH SIZE ON LOSS

Batch Size	Accuracy
Batch Size	Loss
8	17.84
16	13.97
32	9.03
64	6.21

TABLE 2
IMPACT OF BATCH SIZE ON ACCURACY

Batch Size	Accuracy
8	0.71
16	0.69
32	0.714
64	0.705

TABLE 3
IMPACT OF BATCH SIZE ON ACCURACY

Batch Size	Accuracy	Loss	Dropout	# of Step	# of Epoch
8	0.706	27.6	0.5	13500	200
16	0.71	16.5	0.5	67600	200
32	0.701	9.9	0.5	33800	200
64	0.72	6.2	0.5	17000	200

ISSN: 2414-4592
DOI: 10.20474/jater-2.5.3

In our work we try also tuning the hyperparameter dropout to examine our model to get the better performance. In the beginning start with the value 0.1 and test the network for different batch size as show in table 3.

And later on, we try tuning dropout value and make it equal 0.5 at check the result regarding to the accuracy as shown in table 4. However, after testing our network on more time we get the best performance at batch size =64 and dropout=0.5. Finally, we experiment our network by keeping the batch size 64 and tuning the dropout between the intervals [0.1-0.5] as shown in table 5. We get a minimum loss and better accuracy at batch size 64 and dropout value equal 0.5. To sum up, we put all the result in one table as depict in table 6.

VIII. CONCLUSION AND FUTURE WORK

In the presented work, we performed an experimental test on CNN built on learning word embedding from scratch for sentence classification. By tuning hyperparameter of CNN we get better performance in term of accuracy and regularization. For future work, we can start from the embedding with pre-trained word2vec vectors.

TABLE 4
THE RESULT AFTER APPLYING MULTIPLE BATCH SIZE AND FIXED VALUE FOR DROPOUT =0.5

Batch Size	Accuracy	Loss	Dropout	# of Step	# of Epoch
8	0.71	17.84	0.1	13500	200
16	0.69	13.97	0.1	67600	200
32	0.714	9.03	0.1	33800	200
64	0.705	6.21	0.1	17000	200

TABLE 5
ILLUSTRATE THE RESULT OF MULTIPLE DROPOUT VALUE AND THE ACCURACY

Batch Size	Dropout	Accuracy	Loss
64	0.705	6.21	0.1
64	0.69	6.5	0.2
64	0.72	6.3	0.3
64	0.718	6.5	0.4
64	0.72	6.2	0.5

TABLE 6
ILLUSTRATE ALL THE CASE OF THE HYPERPARAMETERS VALUE

Batch Size	Accuracy	Loss	Dropout	# of Step	# of Epoch
8	0.706	27.6	0.5	13500	200
16	0.71	16.5	0.5	67600	200
32	0.701	9.9	0.5	33800	200
64	0.72	6.2	0.5	17000	200
64	0.705	6.21	0.1	17000	200
64	0.69	6.5	0.2	17000	200
64	0.72	6.3	0.3	17000	200
64	0.718	6.5	0.4	17000	200

ISSN: 2414-4592
DOI: 10.20474/jater-2.5.3

REFERENCES

[1] Y. Bengio, R. Ducharme, P. Vincent and C. Jauvin, "Neural probabilistic language model," *Journal of Machine Learning Research* 3, pp. 1137-1155, 2003.

[2] Y. Zhang and B. Wallace, "A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification," 2015. [Online]. Available: arXiv:1510.03820

[3] Y. Kim, "Convolutional neural networks for sentence classification," 2014. [Online]. Available: arXiv:1408.5882

[4] X. Zhang, J. Zhao and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in Neural Information Processing Systems*. San Francisco, CA: Morgan Kaufmann Publishers Inc. 2015, pp. 649-657.

[5] P. Wang, J. Xu, B. Xu, C. L. Liu, H. Zhang, F. Wang and H. Hao, "Semantic clustering and convolutional neural network for short text categorization," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, 2015, pp. 352-357. **DOI:** 10.3115/v1/p15-2058.

[6] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2011, pp. 142-150.

[7] B. Pang, L. Lee and S. Vaithyanathan, "Thumbs up?: Sentiment classification using machine learning techniques," in *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing*, 2002, pp. 79-86. **DOI:** 10.3115/1118693.1118704.

[8] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, 1998. **DOI:** 10.1109/5.726791.

[9] R. Johnson and T. Zhang, "Effective use of word order for text categorization with convolutional neural networks," 2014. [Online]. Available: arXiv:1412.1058.

[10] T. Mikolov and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems.* San Francisco, CA: Morgan Kaufmann Publishers Inc. 2015.

[11] Y. L. Boureau, J. Ponce and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *Proceedings of the 27th International Conference on Machine Learning*, 2010, pp. 111-118.

[12] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," 2012. [Online]. Available: arXiv:1207.0580

[13] D. Britz, "Implementing a CNN for text Classification in TensorFlow," 2016. [Online]. Available: goo.gl/JnoVkh

[14] "CS231n convolutional neural networks for visual recognition," 2016. [Online]. Available: goo.gl/rFQjcC

— This article does not have any appendix. —

ISSN: 2414-4592
DOI: 10.20474/jater-2.5.3