Deep Learning Approach for Sentiment Analysis of Short Texts

Abdalraouf Hassan
Dept. of computer science and engineering
University of Bridgeport
Bridgeport, CT
e-mail: abdalrah@my.bridgeport.edu

Ausif Mahmood
Dept. of computer science and engineering
University of Bridgeport
Bridgeport, CT
e-mail: mahmood@bridgeport.edu

Abstract—**Unstructured text data produced on the internet grows rapidly, and sentiment analysis for short texts becomes a challenge because of the limit of the contextual information they usually contain. Learning good vector representations for sentences is a challenging task and an ongoing research area. Moreover, learning long-term dependencies with gradient descent is difficult in neural network language model because of the vanishing gradients problem. Natural Language Processing (NLP) systems traditionally treat words as discrete atomic symbols; the model can leverage small amounts of information regarding the relationship between the individual symbols. In this paper, we propose *ConvLstm*, neural network architecture that employs Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) on top of pre-trained word vectors. In our experiments, *ConvLstm* exploit LSTM as a substitute of pooling layer in CNN to reduce the loss of detailed local information and capture long term dependencies in sequence of sentences. We validate the proposed model on two sentiment datasets IMDB, and Stanford Sentiment Treebank (SSTb). Empirical results show that *ConvLstm* achieved comparable performances with less parameters on sentiment analysis tasks.**

Keywords-convlutional neural network; long short-term memroy; recurrent neural network

I. INTRODUCTION

Text classification is an essential task for Natural Language Processing (NLP) with many applications, such as information retrieval, web search, ranking and spam filtering, in which we need to assign single or multiple predefined categories to sequence of text. The objective of NLP is to process the text in order to analyze it and extract information to represent it differently. Today text classification research starts from designing the best feature extractors to choosing the best possible machine learning classifiers. The classic approach to text classification typically starts with feature extraction stage then is followed by a classifier stage. Perhaps one popular technique is to represent a sentence as bag-of-words, then train a linear classifier, (e.g., a logistic regression or SVM) [1, 2].

Recently, models based on Neural Networks have become increasingly popular [3]; it has become possible to train more complex models on much larger dataset. They typically outperform the simple models. Perhaps the most efficient model is to use distributed representation of word [4]. For instance neural network language models outperform N-gram models [5, 6]. Currently for text classification problems a linear classifiers are considered to be the conventional approach and strong Baselines [1]. Regardless of their simplicity, the linear classifier often obtains the state-of- art performances especially when the correct features are selected.

Deep neural network methods jointly implement feature extraction and classification for document classification [3, 7]. The deep neural network based approach convention, in most cases, is an input document represented as a sequence of words, and each sequence is then represented as one-hot-vector, each word in the sequence is projected into a continuous vector space by multiplying it with weight matrix, forming a sequence of dense, real valued vector. This sequence is then fed into a deep neural network, which processes the sequence in multiple layers, finally resulting in prediction probability. This pipeline is tuned jointly to maximize the classification accuracy on training set.

Convolutional Neural Network (CNN) has recently accomplished a remarkable performance on the essentially significant task of sentence classification [3, 8, 9]. However, these models require professionals to specify an exact model architecture and set accompanying hyper-parameters, including the filter region size. Recent work by [10] consists of multi layers of CNN and max pooling, similar to the architecture proposed by [11] in computer vision. In the first stage, each layer will extract features from small overlapping windows of the input sequence and pools over small non-overlapping windows by taking the maximum activation in the window. This is applied recursively for many times. The final convolutional layers are then flattened to form a vector, which feeds into a small number of fully connected layers followed by a classification layer.

We observed that employing convolutional to capture long term dependencies requires many layers, because of the locality of the convolutional and pooling. As the length of the input grows, this become crucial; that was the motivation behind [10] to investigate deep convolutional network to overcome these issues. [12] investigated the combination of neural network architecture of CNN and Recurrent Neural Network (RNN) in order to encode character input, which was implemented to learn high-level feature input sequences of character level to capture sub word information. However, this model performs better only when a large number of classes are available. Another successful model applied RNN for NLP was introduced by [13, 14]; it confirmed that RNN is able to capture long-term dependencies even in the case of a single layer. Today RNN is the lead approach for many NLP applications. Recursive Neural Network was applied to sentence classification [15]; configuration function is defined in this model and recursively applied at each node of the parse tree of an input sentence. In order to extract a feature vector of the sentence, the model relies on an external parser.

In this paper, we propose a neural language model that leverages both convolutional and recurrent layers to efficiently perform text classification tasks. Based on our observation from the work proposed in [10] CNN architecture must have many layers to capture long-term dependencies in an input sentence. Our work is also inspired from the fact that recurrent layers are able to capture long-term dependences with one single layer [14]. In our model, we utilize a recurrent layer LSTM as substitutes for the pooling layer in order to reduce the loss of detailed local information and capture long-term dependencies. Surprisingly, our model achieved comparable results on two sentiment analysis benchmarks with less number of parameters. We show that it is possibly to use a much smaller model to achieve the same level of classification performance when recurrent layer combined with convolutional layer.

II. Related Work

Deep Learning achieved significant results in computer vision [11] and speech recognition [22]. It has become more common to use deep neural methods in NLP applications; much of the work has involved learning word vector representations through neural language models, then performing composition over the learned word vectors for classification [23]. Deep neural networks and representation learning approaches have led to new methods for solving the data sparsity problem. Several neural network based models for learning word representations followed this approach. Word embedding is the neural representation of a word and is a real vector. Word embedding allows us to measure similarity between words by simply using the distance between two embedded vectors [23, 24]. Recently, researchers observed that is not necessary for deep neural network to perform at word level. As long as the document represented as one-hot-vector, the model could work without any change, regardless if each one-hot vector corresponds to a word [3]. Character sequence proposed as an alternative to the one-hot vector [25] . Similar ideas also applied to dependency parsing in [26]. Deep CNN for NLP by [27] composed numerous of layers of convolutional and max pooling, it is identical to the convolutional architecture in the computer vision [11].

CNN was initially designed for computer vision and exploits layers with convolving filters that are applied to local features. CNN reached an outstanding result in computer vision where handcrafted features were used, e.g. scale-invariant features transform (SIFT) followed by a classifier. The main idea is to consider features extractors and classifier as one jointly trained task [21]. The use of neural networks inspired many researchers after the successful approach in [23]. CNN models for NLP achieved excellent results in semantic parsing [28], sentence modeling [9], search query retrieval [14], and other NLP tasks [23].

Our technique is inspired by the recent work with deep neural networks for NLP proposed in [3, 10, 12, 20]. RNN applied in [12] to support the convolutional layers to capture long-term dependencies across the whole document. The combination of CNN and RNN also explored for image segmentation tasks [16, 17]. Similar work was applied to speech recognition [18]. Convolution-gated recurrent network is perhaps the most relevant work [19]. A hierarchical processing of document and bidirectional RNN are appointed to extract the feature vector of document. This vector is mapped into the classification layer. The RNN focused to model inter-sentence structures, and the convolutional firmly forced to model each sentence.

In this work, we propose a neural language model *ConvLstm,* which utilizes both convolution and recurrent layers on top of pre-trained vectors. Pre-trained word vectors were obtained from unsupervised neural language model [24], and it is able to capture syntactic and semantic information among word representations. The convolutional layer can extract high-level features from input sequence efficiently. However, it requires many layers of CNN to capture long-term dependencies [10]. The recurrent layer LSTM has the ability to remember important information across long stretches of time and also is able to capture long-term dependences [14]. Our work is inspired by [3, 10, 12]. Based on these observations, the proposed model combines a convolutional and a recurrent layer on top of word2vec (unsupervised pre-training of word vectors is an important ingredient in deep learning for NLP). Then we employ a recurrent layer as an alternative for pooling layers to efficiently capture long-term dependencies for the text classification tasks. Our model achieved a competitive results on multiple benchmarks with less number of parameters.

III. Neural Network Architececure

A. *Convolutional Neural Network*

The model shown in Figure 1 is a slight variant of the CNN architecture of [23]. Le$x_i \in \mathbb{R}^k$ to be the k-dimensional word vector corresponding to the $i\text{-}th$ word in the sentence. A sentence of length n padded where necessary is represented as:

$$x_1 = x_1 \oplus x_2 \oplus \dots \oplus x_n, \qquad (1)$$

where $\oplus$ is the concatenation operator, and the convolutional operational consists of a *filter* $w \in \mathbb{R}^{hk}$, which is applied to a window of h words to produce a new feature. For instance, feature c_i is generated from a window of words $x_{i:i+h-1}$ by:

$$c_i = f(w.x_{i:i+h-1} + b). \qquad (2)$$

Where $b \in \mathbb{R}$ a bias is a term and f is a nonlinear function (e.g. rectifier or tanh). This is done for every time step of the input sequence $\{X_{1:h}, X_{2:h+1}, \dots, X_{n-h+1:n}\}$ to produce a feature map of:

$$c = [c_1, c_2, \dots, c_{n-h+1}], \qquad (3)$$

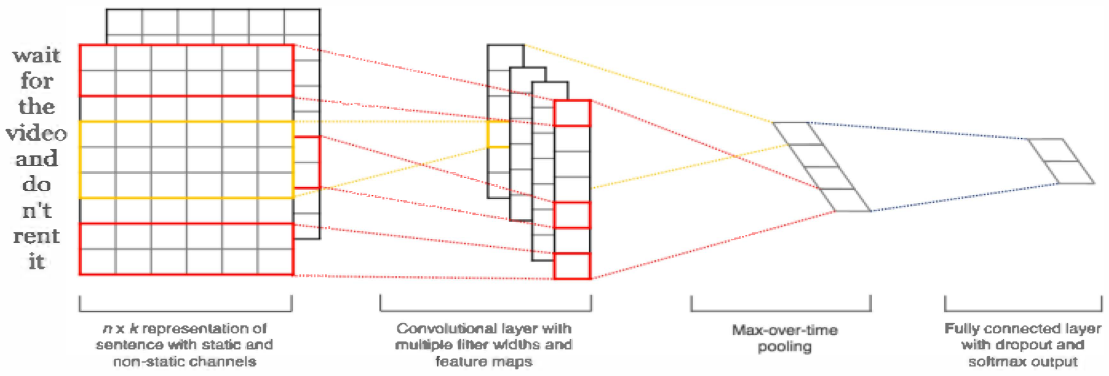
wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Figure 1. CNN Model architecture for NLP.

Max-over-time pooling operation was not applied; we fed the feature map into single layer of LSTM.

B. Recrrent Neural Network

The objective of the RNN is to make use of sequential informations, and the output is based on the previous computation. All inputs are independent of each other in traditional neural network, while this approach is inefficient for many tasks in NLP (e.g. predicting the next word in a sentence) in this case it is important to know the previous word. RNN has a memory that capture informations in arbitrary long sequences, which is illustrated in Fig. 2.

$$h_t = f(x_t, h_{t-1}), \tag{4}$$

where $x_t \in \mathbb{R}^d$ one time step from the input sequence, $(x_1, x_2, \dots, x_T)$. $h_0 \in \mathbb{R}^{d`}$ often initialized as an all-zero vector. Recursive neural networks proved to be efficient for constructing sentence representations. The model has a tree structure which is able to capture the semantic of sentence. However, this is a time-consuming task due to constructing the textual tree complexity [29]. Recurrent neural network has enhanced time complexity. In this model, text is analyzed word by word and then preserves the semantic of all the previous text in a fixed-sized hidden layer [30]. The capability to capture superior appropriate statistics could be valuable for capture semantics of long text in recurrent networks. However, recurrent networks is biased model, because recent words are more significant than earlier words. Therefore, the key components could appear anywhere across the document and not only at the end; this might reduce the efficiency when used to capture the semantic of whole document. The LSTM model was introduced to overcome these difficulties.

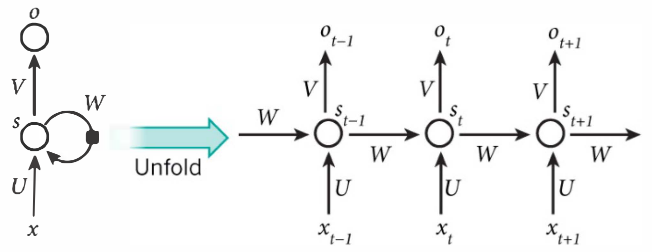

Figure 2. RNN unfold framework.

The most naive recursive function is known to suffer from the problem of vanishing gradient. More recently it is common to use Long Short-Term Memory LSTM [31, 32].

C. Long Short-Term Memory

LSTM is more complicated function that learns to control the flow of information, to prevent the vanishing gradient and to allow the recurrent layer to more easily capture long-term dependencies. LSTM was initially proposed in [31, 32] and later modified in [33]. RNN has problems of gradient vanishing or explosion. Meanwhile, RNNs are considered as deep neural networks across many time instances. The gradient at the end of the sentence may not be able to back-propagate to the beginning of the sentence, because of the nonlinearity transformation [11, 30].

These problems are the main motivation behind the LSTM model, which introduces a new structure called a memory cell in figure 3.

The memory cell is consist of four main components: input, output, forget gates and candidate memory cell.

The following equations describe how the memory cells layer are updated at every timestep t. First, we compute the values for i_t, the input gate, and $\tilde{c}_t$ the candidate value for the states of the memory cells at time t:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{5}$$

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{6}$$

Secondly, we compute the value for f_t, the activation of the memory cells forgets at time t:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{7}$$

Given the new value of the input gate activation i_t, the forget gate activation f_t and the candidate state value $\tilde{c}_t$, we can compute c_t the memory cells new state at time t:

$$c_t = i_t * \tilde{c} + f_t * c_{t-1} \tag{8}$$

With the new state of the memory cells, we compute the value of their output gates and, subsequently, their outputs:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o c_t + b_o) \quad (9)$$

$$h_t = o_t * \tanh(c_t) \quad (10)$$

Where x_t is the input to the memory cell layer at time t. $W_i, W_f, W_c, W_o, U_i, U_f, U_c, U_o$, and V_o are weight matrices. b_i, b_f, b_c, b_o, are bias vectors.

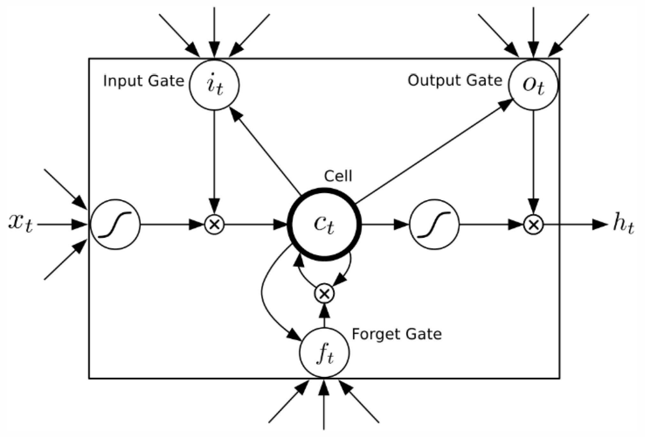

Figure 3. LSTM framework shows the network using only the last hidden dimension for sentiment prediction.

D. Unsupervised Learning of Word-Level Embedding

To improve the performance of the model with the absence of a large supervised training set; initializing the word vectors with pre-trained vectors obtained from an unsupervised neural language model is a very successful method [3]. It can capture syntactic and semantic information, which are very important to sentiment analysis. In our work, we preform unsupervised learning of word-level embeddings using word2vec tool [20], that implement skip-gram and continuous bag-of-words architectures for computing vector representations of words.

IV. DATASETS

To evaluate the performance of our model, we used two sentiment analysis datasets, Stanford Large Movie Review Dataset IMDB [34] and Stanford Sentiment Treebank dataset SSTb [35], as shown in Table I.

A. IMDB Sentiment Analysis Dataset

We benchmark our model on the IMDB movie sentiment dataset, proposed by [34] .The Stanford Large Movie Review Dataset consists of 50,000 binary labeled reviews from IMDB. The reviews are divided into 50:50 training and testing sets. The spreading of labels within each subsection of data is composed. The IMDB dataset includes 50,000 unlabeled reviews that might be used for unsupervised training. In the first stage of our experiments, we use 15% of the labeled training documents as a validation set. The average length of each document is 241 tokens, with standard deviation of 198.8 tokens; the maximum length of a document is 2,526 words.

B. Stanford Sentiment Treebank Dataset

Stanford Sentiment Treebank (SSTb) is an extension of the MR Dataset, but with train/dev/test splits provide and fine-grained label introduced by [35]. It consists of 11855 reviews from Rotten Tomatoes, with one sentence for each review. The SSTb dataset was spilt into 8,544 sentences for training, validation or development (dev) 1101, and 2210 sentences for testing.

TABLE I. SSTB, IMDB SENTIMENT ANALYSIS DATASETS.

DATASET	Set	# Sentences	# classes
SSTb	*Train*	*8544*	*5*
	Dev	*1101*	*5*
	Test	*2210*	*5*
IMDB	*Train*	*25k*	*2*
	Dev	*198.8*	*2*
	Test	*25k*	*2*

V. EXPERMENTAL SETUP AND RESULTS

A. Model Settings

Many different combinations of hyper-parameters can give similar results. We devoted extra time tuning the learning rate, dropout and the number of units in the convolutional layer, since these hyper-parameters has a large impact on the prediction performance. The number of epochs varies between (5, 10) for both dataset. We believe that by adding a recurrent layer as a substitute to the pooling layer it can effectively reduce the number of the convolutional layers needed in the model in order to capture long-term dependencies. Therefore, we consider emerging a convolutional and a recurrent layers in one single model *ConvLstm*, with multiple filters width (3, 4, 5), feature maps = 256, for activation functions in the convolutional layer we used rectified linear (ReLus) for nonlinearity, padding was set to zero. All elements that would fall outside the matrix are taken to be zero. To reduce overfitting we applied dropout 0.5 only before the recurrent layer. The main contribution in our work, exploits recurrent layers as substitutes for the pooling layer; a key aspect of CNNs are pooling layers, which are typically applied after the convolutional layers to subsample their input, a max operation is the most common way to do pooling operation. However, we removed it in our model and used recurrent layer instead, the recurrent layer is fixed to a single layer of LSTM vanilla architecture [36, 37]. We also used gradient clipping [38] to cell output and gradients. Our LSTM has input, forget, and output gates, hidden state dimension is set to 128. The task are sentiment analysis using IMDB and Stanford Sentiment Treebank in Table II and Table III. In our model we set the number of filters in the convolutional layers to be the 2x as the dimension of the hidden states in the recurrent layer, which add 3%-6% relative performance.

B. Regularization

Dropout is an effective way to regularize deep neural networks [3]. We observe applying dropout before and after the recurrent layer that decrease the performance of the model

2%-5%, therefore, we only apply dropout after the recurrent layer, we set the dropout to 0.5. Dropout prevent co-adaptation of hidden units by randomly dropping out.

C. *Training and Validation*

Training is done through stochastic gradient descent over shuffled mini-batches, we randomly split the full training examples into training and validation. The size of the validation is the same as the corresponding test size and is balanced in each class. We train the model by minimizing the negative log-likelihood or cross entropy loss. The gradient of the cost function is computed with backpropagation through time (BPTT). Early stopping strategy is utilized to prevent overfitting. Before training we employed unsupervised learning of word-level embedding using the word2vec [24], which implemented the continuous bag-of-words and skip-gram architectures for computing vector representation of word. These vectors were trained on 100 billion words of Google News, word2vec tool is publicly available. We validate the proposed model on two datasets, considering the difference in the number of parameters. The accuracy of the model does not increase with incensement in the number of the convolutional layers, one layer enough to peak the model [3], more pooling layers mostly lead to the loss of long term-dependencies [12]. Thus, in our model we ignored the pooling layer in the convolutional network and replaced it with single LSTM layer to reduce the loss in local information, single recurrent layer is enough to capture long-term dependencies in the model.

TABLE II. ACCURACY ON SSTB DATASET FOR FINE GRAINED (5-CLASS) AND BINARY PREDICTIONS AT THE SENTENCE LEVEL.

Models	Fine-Grained	Binary
Naïve Bayes [35]	41.0	81.8%
SVMs [35]	40.7	79.4%
RNN [35]	43.2	85.4%
CNN-non-static [3]	48.0	87.2%
CNN-multichannel [3]	47.4	88.1%
RNTN [35]	45.7	85.4%
MV-RNN [15]	44.4	82.9%
Paragraph-Vector [39]	48.7	87.8%
ConvLstm	47.5	88.3%

TABLE III. PERFORMANCE OF OUR METHOD COMPARED TO ERROR RATES ON THE IMDB DATASET REPORTED IN [41]

Models	Error rate
NBSVM-uni [41]	11.71%
SVM-uni [41]	13.05%
SVM-bi [41]	10.84%
Full+Unlabeled+BoW [34]	11.11%
BoW-bnc [34]	12.20%
ConvLstm	11.00%

D. *Result and Discussion*

We perform several experiments to offer fair comparison to recent presented deep learning and traditional methods, as shown in Table II and III. For IMDB dataset the previous baseline is bag-of-words [41] and Paragraph Vectors [39].

Our *ConvLstm* model archives comparable performances with significantly less parameters. We achieved better results compared to convolutional only models; it likely loses detailed local features because of the number of the pooling layers. We assumed that the proposed model is more compact because of the small number of parameters and less disposed to overfitting. Hence, it generalizes better when the training size is limited. It is possible to use more filters in the convolutional layer without changing the dimensional in the recurrent layer, which potentially increases the performance 2%-4% without sacrifice of the number of the parameters.

We observed that many factors affect the performance of the deep learning methods, such as the dataset size, vanishing and exploding of the gradient, choosing the best feature extractors and classifiers is still an open research area. However, there is no specific model fit for all types of datasets.

VI. CONCLUSTION

In this paper, we proposed a neural language model to overcome the shortcomings in traditional and deep learning methods. We propose to combine the convolutional and recurrent layer into a single model on top of pre-trained word vectors; to capture long-term dependencies in short texts more efficiently. We validated the proposed model on SSTb and IMDB Datasets. We achieved comparable results with less number of convolutional layers compared to the convolutional only architecture, and our results confirm that unsupervised pre-trained of word vectors is a significant feature in deep learning for NLP. Also using LSTM as an alternative for the pooling layers in CNN gives the model enhancement to capture long-term dependencies.

It will be remarkable for future research to apply the architecture on other NLP applications such as spam filtering and web search. Using other variants of recurrent neural network as substitutes for pooling layers is also area worth exploring.

REFERENCES

[1] Joachims, T. Text categorization with support vector machines: Learning with many relevant features. in European conference on machine learning. 1998. Springer.

[2] Mu, Y., et al., Event-related theta and alpha oscillations mediate empathy for pain. Brain research, 2008. 1234: p. 128-136.

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

[4] Rumelhart, D.E., G.E. Hinton, and R.J. Williams, *Learning representations by back-propagating errors.* Cognitive modeling, 1988. 5(3): p. 1.

[5] Bengio, Y., et al., Neural probabilistic language models, in Innovations in Machine Learning. 2006, Springer. p. 137-186.

[6] Mikolov, T., et al. Strategies for training large scale neural network language models. in Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on. 2011. IEEE.

[7] Shen, Y., et al. Learning semantic representations using convolutional neural networks for web search. in Proceedings of the 23rd International Conference on World Wide Web. 2014. ACM.

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

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

[10] Zhang, X., J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. in Advances in Neural Information Processing Systems. 2015.

[11] Krizhevsky, A., I. Sutskever, and G.E. Hinton. Imagenet classification with deep convolutional neural networks. in Advances in neural information processing systems. 2012.

[12] Xiao, Y. and K. Cho, Efficient Character-level Document Classification by Combining Convolution and Recurrent Layers. arXiv preprint arXiv:1602.00367, 2016.

[13] Bahdanau, D., K. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate.* arXiv preprint arXiv:1409.0473, 2014.

[14] Sundermeyer, M., H. Ney, and R. Schlüter, *From feedforward to recurrent LSTM neural networks for language modeling.* IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP), 2015. **23**(3): p. 517-529.

[15] Socher, R., et al. Semantic compositionality through recursive matrix-vector spaces. in Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. 2012. Association for Computational Linguistics.

[16] Chen, L.-C., et al., Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform. arXiv preprint arXiv:1511.03328, 2015.

[17] Visin, F., et al., *Reseg: A recurrent neural network for object segmentation.* arXiv preprint arXiv:1511.07053, 2015.

[18] Sainath, T.N., et al., *Deep convolutional neural networks for large-scale speech tasks.* Neural Networks, 2015. 64: p. 39-48.

[19] Tang, D., B. Qin, and T. Liu. Document modeling with gated recurrent neural network for sentiment classification. in Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. 2015.

[20] Mikolov, T., et al., *Efficient estimation of word representations in vector space.* arXiv preprint arXiv:1301.3781, 2013.

[21] He, K., et al., *Deep residual learning for image recognition.* arXiv preprint arXiv:1512.03385, 2015.

[22] Graves, A., A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. in 2013 IEEE international conference on acoustics, speech and signal processing. 2013. IEEE.

[23] Collobert, R., et al., *Natural language processing (almost) from scratch.* Journal of Machine Learning Research, 2011. 12(Aug): p. 2493-2537.

[24] Mikolov, T., et al. Distributed representations of words and phrases and their compositionality. in Advances in neural information processing systems. 2013.

[25] Ling, W., et al., Finding function in form: Compositional character models for open vocabulary word representation. arXiv preprint arXiv:1508.02096, 2015.

[26] Ballesteros, M., C. Dyer, and N.A. Smith, *Improved transition-based parsing by modeling characters instead of words with LSTMs.* arXiv preprint arXiv:1508.00657, 2015.

[27] Conneau, A., et al., Very Deep Convolutional Networks for Natural Language Processing. arXiv preprint arXiv:1606.01781, 2016.

[28] Yih, W.-t., X. He, and C. Meek. Semantic Parsing for Single-Relation Question Answering. in ACL (2). 2014. Citeseer.

[29] Socher, R., et al. Parsing natural scenes and natural language with recursive neural networks. in Proceedings of the 28th international conference on machine learning (ICML-11). 2011.

[30] Elman, J.L., *Finding structure in time.* Cognitive science, 1990. 14(2): p. 179-211.

[31] Hochreiter, S. and J. Schmidhuber, *Long short-term memory.* Neural computation, 1997. 9(8): p. 1735-1780.

[32] Gers, F.A., J. Schmidhuber, and F. Cummins, *Learning to forget: Continual prediction with LSTM.* Neural computation, 2000. 12(10): p. 2451-2471.

[33] Graves, A., *Generating sequences with recurrent neural networks.* arXiv preprint arXiv:1308.0850, 2013.

[34] Maas, A.L., et al. Learning word vectors for sentiment analysis. in Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1. 2011. Association for Computational Linguistics.

[35] Socher, R., et al. Recursive deep models for semantic compositionality over a sentiment treebank. in Proceedings of the conference on empirical methods in natural language processing (EMNLP). 2013. Citeseer.

[36] Graves, A. and J. Schmidhuber, Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Networks, 2005. 18(5): p. 602-610.

[37] Greff, K., et al., *LSTM: A search space odyssey.* arXiv preprint arXiv:1503.04069, 2015.

[38] Pascanu, R., T. Mikolov, and Y. Bengio, *On the difficulty of training recurrent neural networks.* ICML (3), 2013. 28: p. 1310-1318.

[39] Le, Q.V. and T. Mikolov. Distributed Representations of Sentences and Documents. in ICML. 2014.

[40] Dai, A.M. and Q.V. Le. Semi-supervised sequence learning. in Advances in Neural Information Processing Systems. 2015.

[41] Wang, Sida, and Christopher D. Manning. "Baselines and bigrams: Simple, good sentiment and topic classification." Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2. Association for Computational Linguistics, 2012.