Euler Recurrent Neural Network: Tracking the Input Contribution to Prediction on Sequential Data

Fengcheng Yuan[1,2], Zheng Lin[2(✉)], Weiping Wang[2], and Gang Shi[1,2]

[1] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[2] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{yuanfengcheng,linzheng,wangweiping,shigang}@iie.ac.cn

Abstract. Recurrent neural networks (RNNs) achieve promising results on modeling sequential data. When a model produce an effective prediction, we always wonder which inputs are crucial to the specific prediction. Modern RNNs use nonlinear transformations to update their hidden states, which is hard to quantify the contributions for each input to the prediction. Inspired by the Euler Method, we propose a novel framework named Euler Recurrent Neural Network (ERNN) that uses weighted sums instead of nonlinear transformations to update its hidden states. This model can track the contribution of each input to the prediction at each time-step and achieve competitive result with fewer parameters. After quantification of their contributions to the prediction result, we can find the decisive ones among inputs and can also better understand the principle of the models in the prediction process.

Keywords: Recurrent neural network · Interpretability · Sequential data

1 Introduction

When machine learning models achieve surprising performance, we usually want to know how the model works. This answer is very significant, because by understanding the principle of the models, we can achieve better performance [13] and make more trustworthy judgments [21]. For risk control, we need further explanations for the model's applications in the fields like medical diagnosis [8] and autopilot [3].

One approach to interpreting neural networks is tracing the models' predictions back to the training data via influence functions [13]. Another work on interpreting neural networks focus on how a fixed model leads to particular predictions, such as using local explanation vectors to find the most influential features [2], locally fitting a simpler model to assert trust for a prediction or a model [21], and perturbing the test data to see how the prediction changes [1,22]. These work focus on feed-forward networks. However, recurrent neural networks may have their own characteristics.

T. Gedeon et al. (Eds.): ICONIP 2019, CCIS 1143, pp. 738–748, 2019.
https://doi.org/10.1007/978-3-030-36802-9_78

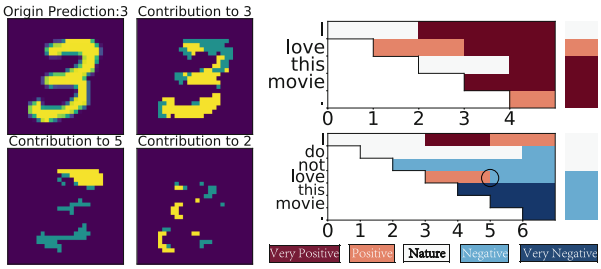

Fig. 1. (a) Explaining a handwritten digits classification made by ERNN. ERNN processes each image one pixel at a time and finally predicts the label. At the final time-step, the top 3 predicted classes are '3' ($p = 0.99$), '5' ($p = 1 \times 10^{-5}$) and '2' ($p = 5 \times 10^{-6}$). Every point has a probability distribution of classes. Points contributed to the class '3' is similar to the original image. (b) An example in sentiment analysis task. ERNN reads one word at a time. The left single column is the prediction at each time-step. Rectangles in two upper triangular matrices indicate the classification that the word contributes to at each time-step. The classification that 'love' contributes to changes from 'Positive' to 'Negative' at time 5 because of the phrase 'do not'.

In recurrent neural networks, Karpathy [11] analyzed an Long Short-Term Memory (LSTM) [9] trained on a character-based language modeling task and broke down its errors into classes, such as "rare word" errors. Murdoch [18] tracked the importance of a given input to the LSTM for a given output by telescoping sums of statistics. Sussillo [25] used nonlinear dynamical systems theory to understand RNNs by solving a set of simple but varied tasks. Hupkes [10] tried to understand the hierarchical compositionality of meaning in gated RNNs with diagnostic classification. However, it is still a challenging problem to quantify the inputs contributions to predictions in RNN models.

In this paper, we focus on the tasks of sequence classification, and further investigate these two tasks by solving one specific problem: what is the contribution of each input to the prediction? Since popular recurrent neural networks use the nonlinear transformation to update states, it is difficult for them to answer this question. To the best of our knowledge, we find that most of the previous work regarded the nonlinear transformation to update hidden states as an essential component of a recurrent neural network [5,26]. A similar work [6] used affine transformations instead of nonlinear transformations to update the hidden states. For each specific input, the model has an affine transformation depending on it. But it is difficult for the model to solve the problem with continuous values inputs.

Inspired by the Euler Method in numerical integration which uses linear additions to update states, we propose a novel framework named Euler Recurrent Neural Network (ERNN) that uses weighted sums instead of nonlinear transformations to update the hidden states. The gate units regulate the information flowing [7]. Based on this description, we view the gating values as weights. The proposed model sums up the contributions of the inputs as its hidden states,

and the weight is dynamically generated by nonlinear functions similar to the gate activation functions in LSTM. In the problem of sequence classification, our model is competitive with LSTM and GRU [4], but with fewer parameters. At the same time, our approach can track the contribution of each input to the prediction at each step. Examples are illustrated in Fig. 1. By quantifying the contributions of the inputs, we can find the key inputs and effectively understand the behavior of the model's prediction.

2 Methods

2.1 Model Definition

In a sequence classification problem, we are given training points $z_1, ..., z_n$, where z_k consists of a sequence of inputs $x_1^{(k)}, x_2^{(k)}, ..., x_t^{(k)}$ and a target $y^{(k)}$. The model read an input x_i at each time-step. Our goal is to reach the target $y^{(k)}$ at the last time-step.

Inspired by the Euler Method which uses addition to update their states, our framework is defined as $\tilde{h}_t = f(x_t), h_t = \gamma_t h_{t-1} + \lambda_t \tilde{h}_t$, where λ is the forget gate, and γ is the input gate. λ and γ can be generated by a trained function, such as Restricted Boltzmann Machine (RBM) [23]. f is an input function, such as RBM or Convolutional Neural Networks (CNN) [15] etc.

Based on the proposed framework, we give two implementations. ERNN-O is a simple version, where forget gate γ_t and input gate λ_t separately depend on h_{t-1} and x_t,

$$\begin{aligned} &\tilde{h}_t = \mathsf{tanh}(W_i x_t + b_i), && h_t = \gamma_t h_{t-1} + \lambda_t \tilde{h}_t \\ &\lambda_t = \sigma(U_\lambda h_{t-1} + W_\lambda x_t + b_\lambda), && \gamma_t = \sigma(U_\gamma h_{t-1} + W_\gamma x_t + b_\gamma). \end{aligned}$$

ERNN-X is a sophisticated version, where forget gate γ_t and input gate λ_t jointly depend on s_t,

$$\begin{aligned} &\tilde{h}_t = \mathsf{tanh}(W_i x_t + b_i), && h_t = \gamma_t h_{t-1} + \lambda_t \tilde{h}_t \\ &\lambda_t = \sigma(U_\lambda s_t + b_\lambda), && \gamma_t = \sigma(U_\gamma s_t + b_\gamma) \\ &s_t = \mathsf{tanh}(U_s h_{t-1} + W_s x_t + b_s). \end{aligned}$$

Here, $W_i, W_\lambda, W_\gamma, W_s \in \mathbb{R}^{n*m}$, $U_i, U_\lambda, U_\gamma, U_s \in \mathbb{R}^{n*n}$, $b_i, b_\lambda, b_\gamma, b_s \in \mathbb{R}^n$. σ denotes the sigmoid activation function, and tanh denotes the hyperbolic tangent activation function. In the experiments, we find that ERNN-X can effectively deal with data with longer sequence.

2.2 The Inputs Contributions to Hidden States and Predictions

Let $h_0 = 0$, the general formula of the Euler Recurrent Neural Network is

$$h_t = \sum_{k=1}^{t} \underbrace{\prod_{i=k+1}^{t} \gamma_i}_{\Gamma_{k+1}^t} \lambda_k \underbrace{f(x_k)}_{f_k} \triangleq \sum_{k=1}^{t} \underbrace{\Gamma_{k+1}^t \lambda_k f_k}_{C_k^t} \triangleq \sum_{k=1}^{t} C_k^t,$$

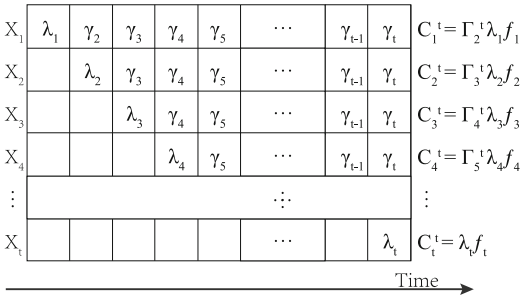

Fig. 2. The input contribution C_k^t to hidden state change dynamically. The vertical direction is the input in time series. Once a new input arrives, the previous inputs contributions will be changed by the forget gate.

where $C_k^t = \Gamma_{k+1}^t \lambda_k f_k$ is the contribution of the input x_k to the hidden state h_t. $\Gamma_{k+1}^t \triangleq \prod_{i=k+1}^t \gamma_i$ is the **forget factor** of the input x_k. Thus the hidden state h_t is the sum of a series of input contributions. Figure 2 illustrates that the contributions of the inputs to the hidden states vary dynamically. When a new input x_k comes, the inputs before time k must multiply the forget gate γ_k. Therefore, the contribution C_k of the input x_k varies with time.

For sequence classification problem, the probabilities are computed as

$$p_t = \mathsf{softmax}(l_t),\ l_t = W_a h_t = W_a \sum_{k=1}^t C_k^t,$$

where l_t is an unnormalized vector, which is used to track the input contribution to prediction. Specifically, the contribution of the input x_k to the prediction at time t is $W_a C_k^t$. Overall, it is the weighted sum updating method that allows us to track the contributions of inputs to predictions easily.

2.3 Relation to Long Short-Term Memory

In LSTM, the nonlinear gating units regulate the information flow into and out of the cell [7]. Our work is heavily influenced by the gating mechanism in LSTM. Obviously, the input gate and the forget gate in ERNN are similar to the ones in LSTM. All gates use nonlinear functions to compute a value. Compared with LSTM, the first difference is that ERNN only use weighted sums to update the hidden state. Another difference between the two models is that we do not use the output gate in ERNN, which makes our model more concise. The form of LSTM is as follows:

$$\begin{aligned} f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f),\ i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \\ o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o),\ \tilde{c}_t = \mathsf{tanh}(W_c x_t + U_c h_{t-1} + b_c) \\ c_t &= f_t c_{t-1} + i_t \tilde{c}_t, \qquad\qquad h_t = o_t \mathsf{tanh}(c_t), \end{aligned}$$

where f_t is forget gate, i_t is input gate and o_t is output gate. To investigate the impact of the first difference, we only remove the hidden-hidden updating matrix U_c in LSTM and name it LSTM-h, $\tilde{c}_t = \mathsf{tanh}(W_c x_t + b_c)$.

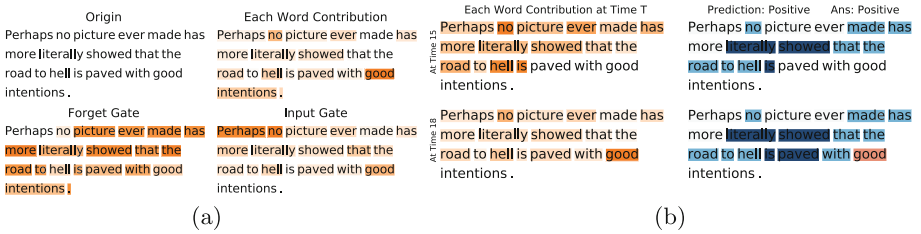

Fig. 3. An example in SST corpus. (a) Contributions of words to *the last hidden state* and gates. (b) Words contributions and predictions change dynamically.

In the experiments, we find that the LSTM-h makes a slight improvement over the LSTM on sentiment analysis task and handwritten digits classification (MNIST) task, though it has a more concise form. The impact of the second difference can be investigated by comparing ERNN with LSTM-h.

3 Experiments and Analysis

To assess the ability of ERNN on tracking the inputs contributions, we test our model on four tasks: sentiment analysis and handwritten digits classification (MNIST) task. Then we analyze how inputs influence the prediction, and discuss the relationship between two gates and inputs. We also compare our models with several popular RNNs, such as GRU and LSTM. The code for replicating our experiments is available on GitHub.

3.1 Sentiment Analysis

Task. We use three datasets, IMDB [16], Movie reviews (MR) [19] and Stanford Sentiment Treebank (SST) [24]. Data in IMDB and MR is binary labeled ('Positive' or 'Negative'). SST contains fine-grained labels ('Very Positive', 'Positive', 'Neutral', 'Negative', 'Very Negative'). In SST and MR corpus, we follow the experimental setup by Kim [12]. In IMDB corpus, we follow the standard data split rules and set aside 5,000 training samples for validation purposes.

Model Setting. All models have 200 hidden units and are trained with 300-dimension pretrained word vector[1] [17]. The pretrained word vector will not be modified during training. Models are trained for 150 epochs to minimize cross-entropy loss. Gradient clipping [20] with a threshold of 5 is applied to the loop variables. Training is performed with Adam on batches of 64. The dictionary size on IMDB and MR is 10000, and for SST we set the dictionary size as 20000. We train three models as our baseline methods: RNN, LSTM, and GRU.

[1] https://code.google.com/archive/p/word2vec/.

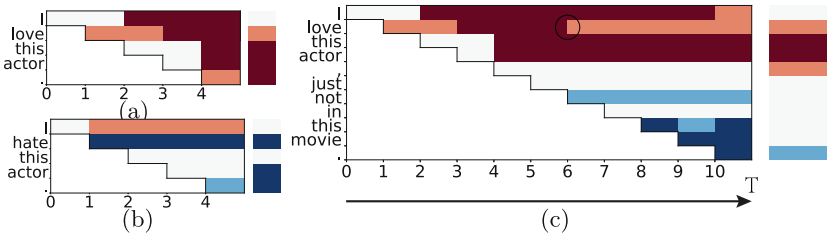

Fig. 4. Contributions of words to the *predictions*. We track every words at middle time-steps. The left single column is the prediction at time-step t. Deep red, red, white, blue and deep blue indicate 'Very positive', 'Positive', 'Nature', 'Negative' and 'Very Negative' respectively. Note images are the upper triangular matrix. (a) The word 'love' has an influence on the following words and the first words. (b) The word 'hate' affect the following words, which is obviously different from the previous sentence. (c) The contribution of the word 'love' to prediction changes when the word 'not' arrives. (Color figure online)

Case Study. We test ERNN on two sentences and explain the model in three aspects. What's the relationship between two gates and words? Fig. 3(a) illustrates that the forget gate has a small value at words 'no', 'hell', 'good' and the first word in the sentence. Then it maintains a large value at other words. The input gate has a larger value at words 'no', 'good' and the first one. Figure 3(b) illustrates that the sentiment of this sentence is negative at time-step 15. When the word 'good' appears, the value of forget gate becomes smaller (means the previous information needs to be forgotten). Thus, the contribution of the word 'no' gets smaller. Eventually, the word 'good' makes the greatest contribution. Figure 3 illustrates that two gates pick the keywords in this sentence.

How does the input gate affect the following words? To get a better understanding of the contributions of the words to the *classification*, we input another three sentences. Note that we only compare the vector l_t before softmax function. In Fig. 4, three input sentences are: 'I love this actor.', 'I hate this actor' and 'I love this actor, just not in this movie.' The only difference between the input sentences of (a) and (b) is the second word, 'love' and 'hate'. And this difference affects the contribution of the following word to the classification and the subsequent predictions. In Fig. 4(a) and (b), the contributions of the word 'this', 'actor' and '.' to the predicted classes are significantly different. The word 'this' contributes to the class 'Very Positive' instead of 'Nature' when the word 'love' appears. Figure 1(b) illustrates that the phrase 'do not' affects the contributions of the subsequent words such as 'this' and 'movie'. Affected by the word 'love','movie' contribute to the class 'Very Positive'. However, affected by phrase 'do not love', 'movie' contributes to the class 'Very Negative'. Figure 1(b) and Fig. 4(a, b) illustrate that the input gate can affect the contributions of the subsequent words.

Table 1. Classification accuracies for models on various data sets. The average and standard deviation results are reported from 10 trials.

Models	IMDB	MR	SST	MNIST
RNN	67.34 ± 10.3	66.56 ± 9.36	35.35 ± 3.10	–
LSTM	90.05 ± 0.30	77.54 ± 0.53	47.26 ± 0.72	89.33*
LSTM-h	**90.27 ± 0.28**	78.31 ± 0.63	47.93 ± 1.06	**98.49 ± 0.12**
GRU	90.14 ± 0.25	**78.34 ± 0.81**	47.51 ± 0.80	98.44 ± 0.28
ERNN-O	86.06 ± 2.82	77.95 ± 0.39	**48.27 ± 0.75**	88.63*
ERNN-X	90.00 ± 0.21	78.28 ± 0.61	47.48 ± 0.71	98.22 ± 0.14

How does the forget gate affect the previous word? A more complex example is shown in Fig. 4(c). When ERNN reads the first 4 words, it gives the same prediction as (a). However, affected by the input word 'not', the contribution of word 'love' to classification change from 'Very Positive' to 'Positive'. Figure 4(c) illustrates that the forget gate can changes the contribution of the previous word.

Analysis. Table 1 illustrates that ERNN achieves competitive results with fewer parameters on the three datasets. The average length of the reviews in IMDB is approximately 240 words, which is larger than the average length of the MR and SST datasets. We find that ERNN-O has larger standard deviation on modeling long sentences (IMDB). Meanwhile, ERNN-O is good at modeling short sentences. It achieves the best result at SST dataset. ERNN-X performs well on three datasets. Impressively, LSTM-h always achieves a better result than LSTM on all these datasets. Thus, the hidden-hidden matrix U_c may not be the key part of LSTM for the sentiment analysis task.

3.2 MNIST

Task. We evaluate our models on a sequential version of the MNIST handwritten digits classification task [15]. The model processes one pixel at a time and finally predicts the label. By flattening the 28×28 images into 784-d vectors, it can be reformulated as a challenging task for RNNs where long-term dependencies need to be leveraged [14].

Model Setting. We follow the standard data split rules and set aside 5,000 training samples for validation. All models are trained with 100 hidden units. After processing all pixels with an RNN, the last hidden state is fed into a softmax classifier to predict the digital class. All models are trained for 200 epochs to minimize cross-entropy loss. We train two models as our baselines, LSTM and GRU. The vanilla RNN has a very low accuracy at the test set.

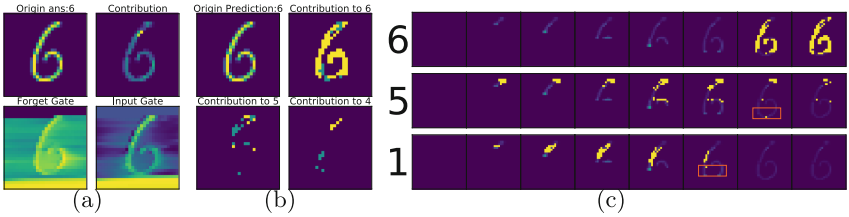

Fig. 5. (a) Contributions and gates. In the second figure, the contribution of each point to *the last hidden state* is similar to the original image. The forget gate and the input gate also show similar phenomenon and contain a large value at the bottom. (b) The contributions of inputs to *predictions*. At last time-step, The top 3 predicted classes are '6' ($p = 0.99$), '5' ($p = 2 \times 10^{-4}$) and '4' ($p = 2 \times 10^{-5}$). (c) We track 3 classes, '6', '5' and '1' at middle time-steps. The number of points that contribute to '5' decrease drastically when the point inside the red square arrives. So does the point that contributes to '1'. But the number of points that contribute to '6' increase in the last few time-steps.

Case Study. We test ERNN with one image (number '6') in test corpus and explain the model in three aspects. What's the relationship between two gates and inputs? Fig. 5(a) shows that the values of the forget gate and the input gate in the pixel of number '6' are larger than the values of other pixels in the image. After encountering the first non-zero pixel, the forget gate maintains a larger value until the last time-step. This means that ERNN-X starts to 'remember' the inputs contributions from the first non-zero pixel. Two gates also have a larger value at lines on the bottom of the image.

Which points contribute to the prediction at the last time-step? In Fig. 1(a), the top 3 predicted classes are '3', '5' and '2'. What ERNN-X picks up for the prediction is consistent with our intuition. Points in the third figure contribute to class '5'. Clearly, the number '5' usually contains a horizontal line. Figure 5(b) shows the contributions of points to predictions at the last time-step. The top 3 predicted classes are '6', '5' and '4'. Points contributed to the class '6' is similar to the original image.

How does the forget gate affect the previous point? We track 3 common predictions (class '6', '5' and '1') in Fig. 5(c). Even though the probability of class '1' at the final time-step is very low, the model has a higher probability of prediction '1' in the first 14 rows. Because points in the first 14 rows are similar to the number '1'. In the third line of Fig. 5(c), when the point inside the red square arrives, the number of points that contribute to '1' decreases sharply, since the forget gate can affect the previous inputs when the model finds that the new input is different from '1'. Thus, points that previously belong to '1' change their contribution classes. In the second line of Fig. 5(c), similar to class '1', the number of points that contribute to '5' drops sharply when the point inside the red square arrives. However, the number of points that contributes to class '6' increases drastically at the last few rows. As we can see, the points contributions to predictions can be changed by the forget gate. By tracking the

input contribution to prediction at middle time-steps, we can better understand the behavior of the model's prediction.

Analysis. Table 1 illustrates that LSTM-h reaches the best result on MNIST. Given the various initial values, we find that LSTM and ERNN-O do not work as well as LSTM-h or GRU. A similar phenomenon on LSTM also appears in Le's work [14]. However, by removing the hidden-hidden matrix U_c in LSTM, LSTM-h alters this phenomenon and outperforms the other models with less parameters. When the output gate in LSTM-h is removed, the model turns out to be ERNN-O. Since ERNN-O performs unsatisfied, we think that the output gate may help leverage long-term dependencies. However, ERNN-X has a different architecture with ERNN-O, and achieves a competitive result with fewer parameters. In ERNN-X, the forget gate and the input gate jointly depending on s_t. This architecture builds strong association between the two gates, which may reduce the fluctuation of the training loss and lead to a better trainable model.

4 Conclusion

In this paper, we propose a novel model named Euler Recurrent Neural Networks which uses weighted sums instead of nonlinear transformations to update the hidden states. This model can track the contribution of each input to the prediction at each time-step. The experiment illustrates that ERNN can not only achieve competitive result with fewer parameters, but also help us better understand the principle of the models in the prediction process.

References

1. Adler, P., et al.: Auditing black-box models for indirect influence. In: 2016 IEEE 16th International Conference on Data Mining (ICDM), pp. 1–10 (2016)
2. Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., Mãžller, K.R.: How to explain individual classification decisions. J. Mach. Learn. Res. **11**, 1803–1831 (2010)
3. Bojarski, M., et al.: End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316, pp. 1–9 (2016)
4. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. In: NIPS 2014 Workshop on Deep Learning, pp. 1–9 (2014)
5. Collins, J., Sohl-Dickstein, J., Sussillo, D.: Capacity and trainability in recurrent neural networks. In: Proceedings of the International Conference for Learning Representations, pp. 1–17 (2017)
6. Foerster, J.N., Gilmer, J., Sohl-Dickstein, J., Chorowski, J., Sussillo, D.: Input switched affine networks: an RNN architecture designed for interpretability. In: Proceedings of the International Conference on Machine Learning, pp. 1136–1145 (2017)
7. Greff, K., Srivastava, R.K., KoutnxEDk, J., Steunebrink, B.R., Schmidhuber, J.: LSTM: a search space odyssey. IEEE Trans. Neural Netw. Learn. Syst. **28**, 2222–2232 (2017)

8. Gulshan, V., et al.: Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. JAMA **316**(22), 2402–2410 (2016)
9. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
10. Hupkes, D., Zuidema, W.: Diagnostic classification and symbolic guidance to understand and improve recurrent neural networks. In: Proceedings of Advances in Neural Information Processing Systems 2017 Workshop, pp. 1–9 (2017)
11. Karpathy, A., Johnson, J., Fei-Fei, L.: Visualizing and understanding recurrent networks. In: Proceedings of the International Conference for Learning Representations 2016 Workshop, pp. 1–12 (2016)
12. Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1746–1751. Association for Computational Linguistics, Doha, Qatar, October 2014
13. Koh, P.W., Liang, P.: Understanding black-box predictions via influence functions. In: Proceedings of the 34th International Conference on Machine Learning, pp. 1885–1894 (2017)
14. Le, Q.V., Jaitly, N., Hinton, G.E.: A simple way to initialize recurrent networks of rectified linear units. arXiv preprint arXiv:1504.00941, pp. 1–9 (2015)
15. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
16. Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning word vectors for sentiment analysis. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pp. 142–150. Association for Computational Linguistics, Portland, Oregon, USA, June 2011
17. Mikolov, T., Corrado, G., Chen, K., Dean, J.: Efficient estimation of word representations in vector space. In: Proceedings of the International Conference on Learning Representations, pp. 1–12 (2013)
18. Murdoch, W.J., Szlam, A.: Automatic rule extraction from long short term memory networks. arXiv preprint arXiv:1702.02540 (2017)
19. Pang, B., Lee, L.: Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In: Proceedings of Association for Computational Linguistics, pp. 115–124 (2005)
20. Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. In: Dasgupta, S., McAllester, D. (eds.) Proceedings of the 30th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 28, pp. 1310–1318. PMLR, Atlanta, Georgia, USA, 17–19 Junuary 2013
21. Ribeiro, M.T., Singh, S., Guestrin, C.: Why should i trust you? explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1135–1144. ACM (2016)
22. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. CoRR abs/1312.6034 (2013)
23. Smolensky, P.: Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chap. Information Processing in Dynamical Systems: Foundations of Harmony Theory, pp. 194–281. MIT Press, Cambridge (1986)

24. Socher, R., et al.: Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 1631–1642 (2013)
25. Sussillo, D., Barak, O.: Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks. Neural Comput. **25**(3), 626–649 (2013)
26. Zhang, S., et al.: Architectural complexity measures of recurrent neural networks. In: Proceedings of Advances in Neural Information Processing Systems, pp. 1822–1830 (2016)