Bulletin of Computer Science and Electrical Engineering
Vol. 1, No. 1, June 2021, pp. 1~8
ISSN: 2722-7324, DOI: 10.25008/bcsee.v1i2.1134

Unfolding Sarcasm in Twitter Using C-RNN Approach

Shawni Dutta[1], Akash Mehta[2]
[1,2]Department of Computer Science, The Bhawanipur Education Society College, Kolkata, India

Article Info

Article history:

Received Nov 29, 2020
Revised Dec 30, 2020
Accepted Mar 27, 2021

Keywords:

Sarcasm Detection
Deep Learning
CNN
RNN
CNN-LSTM
Twitter

ABSTRACT

Sarcasm detection in text is an inspiring field to explore due to its contradictory behavior. Textual data can be analyzed in order to discover clues those lead to sarcasm. A Deep learning-based framework is applied in this paper in order to extract sarcastic clues automatically from text data. In this context, twitter news dataset is exploited to recognize sarcasm. Convolutional-Recurrent Neural network (C-RNN) based model is proposed in this paper that enables automatic discovery of sarcastic pattern detection. The proposed model consists of two major layers such as convolutional layer, and Long-term short memory (LSTM) layers. LSTM is known to be a variant of traditional RNN. Experimental results confirmed sarcastic news detection with promising accuracy of 84.73%. This research work exhibits its uniqueness in combining two dissimilar Deep Learning frameworks under a single entity for predicting sarcastic posts.

Corresponding Author:

Shawni Dutta,
Department of Computer Science,
The Bhawanipur Education Society College,
5, Elgin Rd, Sreepally, Bhowanipore, Kolkata, West Bengal 700020.
Email: shawni.dutta@thebges.edu.in

1. INTRODUCTION

The Sarcasm is a sardonic comment coated by humor. Sarcasm has generally been used to create inconsistencies and uncertainties in the minds of the listeners while being derisive of them or someone else. Sarcasm employs the use of contradiction in order to keep the audience guessing about the true intentions of the host. Sarcasm is generally accompanied by a change in tone, body language and facial expressions while speaking. This makes it easier for sarcasm to be detected in an uttered mode of communication. When it comes to text however, these indicators are absent. Sarcasm detection in texts is done on the basis of contextual information, lexical structures, and use of grammar. This makes Sarcasm detection in texts an interesting task, thereby explaining the immense research interest in them.

Sarcasm is a way of expressing positive feelings using some negative words and phrases, or vice-versa [1]. For example, “You are really smart boy Sheldon #Sarcasm” utters the negative feelings using positive words. Sarcasm is often used as a tool to make jokes, be humorous, or to criticize and make remarks about any product, individual or any proceedings. Different authors have given different definitions of sarcasm. According to [2], the situational differences between the text and the context are often regarded as Sarcasm. Depending on the usage, On the other hand, [3] describes Sarcasm as a pointed and satirical or ironic exclamation designed to cut or give pain. In [3], it is also described that sarcasm can be defined as a mode of satirical wit depending for its

effect on acrimonious, corrosive, and often tongue-in-cheek language that is usually directed against an individual.

The objective of this paper is to recognize sarcastic patterns from news headlines. These news headlines belong to twitter social media platform. This paper focuses on discovering sarcastic patterns from these data. For this purpose, Deep Learning (DL) [4] techniques are utilized while analyzing and inferring sarcasm from tweets. DL techniques are beneficial since it simulates an automated feature extraction method which reduces the burden of manual processing step. DL technique exemplifies the use of neural network model which identifies underlying hidden patterns in the data. Deep neural networks are an improvised version of traditional neural networks in the sense that DNN allows stacking of multiple hidden layers between the input and output layers. Presence of multiple hidden layers will allow learning of features in numerous ways. Recurrent Neural Network (RNN) [5] and Convolutional Neural Network (CNN) [6] follow deep neural network model which is employed in this paper. Long-short term memory (LSTM) [7] is a kind of RNN approach which is exploited in this paper. In addition to it, a major component of the CNN, which is the Convolutional layer, is also used as a part of our proposed methodology. The proposed C-RNN method consists of convolutional layer and LSTM layers. Convolutional layer and Bi-directional LSTM layers [7] are put into a single entity. This implemented method is applied on larger corpus of twitter dataset in order to obtain sarcastic patterns from the dataset.

This section illustrates numerous studies those are dedicated for sarcastic posts detection. The illustration instantiates the necessity of carrying out our current research work. All the mentioned studies are elaborated in terms their employed strategies and efficiencies. Lukin & Walker [8] presented a pattern-based approach that automatically identifies sarcastic and nastiness patterns on unannotated online dialogues. A high precision sarcastic post classifier, followed by a high precision non-sarcastic post classifier is trained using bootstrapping [8] method. Experimental results indicated an accuracy of 68.7% in terms of sarcasm detection. Instead of focusing on feature engineering for extraction of the user-traits, Silvio Amir et al. [9] implemented a CNN [6] for obtaining the contextual features. The past tweets of the user are simply fed to CNN [6], which learns the user characteristics, which is then augmented along with the lexical and syntactical information. González-Ibáñez et al. [10] realized the importance of pragmatic features for ensuring the consistency between positive and negative tweets. They employed SVM [11] with sequential minimal optimization (SMO) and logistic regression (LogR) [12] for differentiating between them. Unlike the approach suggested by González-Ibáñez [10], Barbieri [13] favored the use of features such as punctuations and use of out of context words over patterns of words. The proposed model used seven sets of features such as frequency, synonyms, written-spoken style uses, structural features like length, punctuation, emoticons; concentration of adverbs and adjectives, sentiment gap between confirmatory and toxic terms; and indistinctness. They used a supervised learning algorithm, named the Decision Tree classifier [14] in order to detect sarcasm.

Wang et. al. in [15] tried to make use of the contextual information about the author of the tweet to perceive sarcasm more proficiently. Sifting the tweets through a torrent of posts allowed them to consider a wider context. Three types of contextual information were considered – Topic based Context, History, and Conversation. Two feature engineering methods were used – Word Clusters and Bag of Words [16] – to model the features. These features were then fed into the SVMhnn [15] algorithm for sequential classification. Experimental Results confirmed that sequential classification efficiently detected the contextual information. They were able to demonstrate a significant increase in the performance of the sarcasm detection algorithm. Joshi et al. [17] measured the indirect contextual information those were out of place with its surroundings. These contextual peculiarities have proven to be significant for the detection of sarcasm in texts. This technique [17] detects sarcasm by considering the similarity between word-embedding. The uniqueness of this method lies in the fact that it also takes into account the findings of its predecessors; and augments the features based on the similarity of the word-embedding. The word embedding similarity is calculated using 2 methods – weighted similarity features (WS) and

unweighted similarity features (UWS). They [17] considered four types of word embeddings – GloVe, LSA, Word2Vec, and Dependency Weights. Ghosh and Veale [18] integrated LSTM [7], CNN [6] and a Deep Neural Network [4] in order to identify sarcasm in text. A recursive SVM [11] was created [18] that was provided with labeled syntactic and semantic information, for training. The results generated by the two models [18] were then compared. In the neural network, the text is taken as input and converted into a vector. This vector is then fed into a CNN [6], whose job is to reduce the frequency variation and identify the various discriminating words. These discriminating words are then provided as input into the LSTM [7], which is an RNN [5] capable of extracting temporal contextual information. The output of the LSTM [7] is then fed into a Deep layer, which generates a high order feature set as output. This high order feature set is then sent into a SOFTMAX [19] layer for the final classification. The Neural Network Model [4] was found to outperform the SVM Model [11].

The presented research targets in achieving sarcastic clues detection from news corpus by discarding the need of manual feature engineering task. Our research focuses on automatic sarcastic clue detection using deep learning methodologies because of its self-adaptive nature. The proposed dissimilar neural network model can handle and analyse the news content by itself as well as provide insight to sarcastic pattens present in the news contents.

2. BACKGROUND

Deep Learning (DL) [4] is a widely used machine learning technique which has found its use in applications such as object detection, pattern recognition, and natural language processing. DL belongs to the class of Representation Learning techniques. A DL [4] system has the ability to automatically discover patterns hidden in the raw data. These discovered patterns are then used to perform classification and feature detection. DL [4] is a class of machine learning algorithms which have been inspired by the structure and working of the human brain. A DL [4] system stacks multiple layers of learning-nodes in order to understand the features present in the raw input data. Each layer transforms the output obtained from the previous layer into a representation at a higher and more abstract level. The depth not only allows the system to learn complex features but also enables it to draw inferences which would not have been possible in a shallow system.

Recurrent Neural Networks (RNNs) [5] are a class of Deep Learning methods which was first conceptualized by David Rumelhart in 1986. RNNs [5] are widely used for their ability to properly deal with sequential data. This makes them a perfect tool to deal with Natural Language Processing, Speech Recognition, A/V analysis and Image Captioning. A traditional neural network assumes the data points to be independent of each other, whereas an RNN [5] works really well on sequential data since it captures the time dependencies between the data. What sets RNNs [5] apart from its peers is the ability to share parameters. This characteristic is crucial for it to deal with textual data, since a particular text can be written in multiple ways. Parameter sharing also allows RNNs [5] to deal with variable length sequences, something a traditional multi-layered neural network cannot do. RNNs [5] are an extension of a conventional neural network [4] in the sense that it introduces cycles connecting adjacent nodes in the traditional structure. These cycles work as the internal memory of the entire network. This enables RNNs [5] to deal with past data points. Traditional Neural Networks [4] have a one-to-one mapping between the input and the output, whereas an RNN [5] has a one-to-many, many-to-one, or many-to-many mapping between the input and the output. The recurrent relationship can be denoted by the formula given as equation (1):

$$S(t) = f\big(S(t-1)\big) \tag{1}$$

where S (t) is the state of the system at time t, and S (t-1) is the state of the system at time t-1. Traditional RNNs, also known as Vanilla RNNs, suffer from the problem of exploding or vanishing gradients. This causes the accumulation of errors over multiple time steps. Vanilla RNNs do not

work particularly well when there is a large gap between the referenced data – a common occurrence in texts.

In order to deal with aforementioned problems, variations of RNN were created, one of them being LSTM [7]. LSTM [7] is the most common RNN [5] model that has the ability to remember values over random intervals. It works really well on Time Series Data, and is not affected by the long-term dependency problem which plagued the traditional RNN models. The biggest difference between LSTM [7] and traditional RNN [5] lies in the fact that LSTM [7] has the ability to obtain context from the previous states, in addition to the present states. The vital components of the LSTM [7] are the gates and the memory cells. The working of the LSTM [7] is critically affected by the forget gate, input gate and output gate. The input and forget gates affect the working of the memory cells. If these gates are closed, the memory cell contents remain un-modified between two consecutive time steps. The Gates are responsible for the LSTM's [7] ability to remember information across multiple time-steps, and the flow of the gradient across them. This allows the LSTM [7] to have the ability to be unaffected by the problem ailing the traditional RNN [5] model.

Convolutional Neural Networks (CNNs) [6] are inspired by the working of the human brain – mainly the visual cortex. CNNs [6] are shown to require less parameters compared to its counterparts. The Convolution Layer in the CNN [6] performs Convolutions instead of matrix multiplication. What sets CNN apart are its attributes of parameter sharing and sparse interactions. Parameter sharing is achieved by tying the weights for two different units. Sparse interaction is achieved by having the "kernel" size smaller than the input image. A CNN performs three steps – perform multiple convolutions to generate linear activation, applying nonlinear function on the linear activation, and finally a pooling function that modifies the output of a particular location in the net based on its neighboring values. Examples of pooling functions include – MAXPOOLING, MINPOOLING and AVERAGE-POOLING.

3. RESEARCH METHOD

The target of this paper is to detect whether a given post is sarcastic or not. For this purpose, *News* tweets among which 11724 posts are sarcastic and the rest 14985 are non-sarcastic. Once the data collection is done, we apply a tokenization method as pre-processing techniques on the news *Headlines dataset for Sarcasm Detection* is collected from Kaggle [20]. The dataset consists of 26709 headlines. The words present in the corpus are transformed into lower case for applying pre-processing. The Keras tokenizer API [23] is built on vocabulary size of 10,000 which means a maximum of 10,000 words will be kept based on word frequency. The purpose of this class is to allow the vectorization a text body, by turning each text either into a vector where the coefficient of each token could be binary, based on word count/tf-idf, or into a sequence of integers where each integer is the index of a token in a dictionary.

Next, this tokenizer is fitted into the corpus of tweets and a feature vector is obtained. Later that feature vector is fitted into the proposed classifier model. However, the produced tokenized vector is partitioned into training and testing dataset. Table 1 defines training, testing dataset size, number of sarcastic and non-sarcastic posts. The classifier learns from the training dataset which is given as input in terms of extracted feature vectors. Later, sarcastic pattern prediction results are retrieved using a testing dataset.

Table 1. Distribution of Dataset

Number of Sarcastic tweets	Number of Non-Sarcastic tweets	Number of tweets in Training dataset	Number of tweets in Testing dataset
11724	14985	25,000	1709

After obtaining the pre-processed dataset, it needs to be analyzed for sarcastic pattern identification. To accomplish the objective, the classifier model needs to be employed. Classifier model associates input dataset into target class after discovering hidden relationships among large corpus. This paper utilizes a deep learning framework for implementing a classifier model. RNN and convolutional layer of CNN are the main components of the classifier model. The model consists of one Embedding layer, 1 dimensional convolutional layer, two bi-directional LSTM layers and finally two fully-connected layers respectively.

- **Embedding Layer**: The size of embedding layer is the same as the size of vocabulary size, i.e, 10,000. This layer receives input shape of 40 and dimension of this layer is 2.

- **Convolutional Layer**: This layer is stacked next to the embedding layer. This layer used 1-dimensional convolutional layer which is constructed using a filter size of 32 and kernel size of 3. This layer uses relu [21] as an activation function.

- **Bidirectional LSTM layer**: Following a 1-dimensional convolutional layer, two bi-directional layers are stacked into the model. The layers consist of having learning nodes 64 and 32 respectively. Both these layers are activated using relu [21] function.

- **Flatten Layer**: The output of the last bi-directional LSTM layer produces 3-dimensional output. This layer accepts 3 dimensional inputs and produces 2-dimensional output. This output will be given as input for next fully-connected layers.

- **Fully-connected Layer**: Two fully-connected layers are added into the model of learning nodes 64 and 1 respectively and these layers are activated using the 'relu' [21] and 'sigmoid' [22] activation functions respectively. The last layer is the output layer of the entire model. For designing the fully-connected layers, we employ keras [23] dense layers.

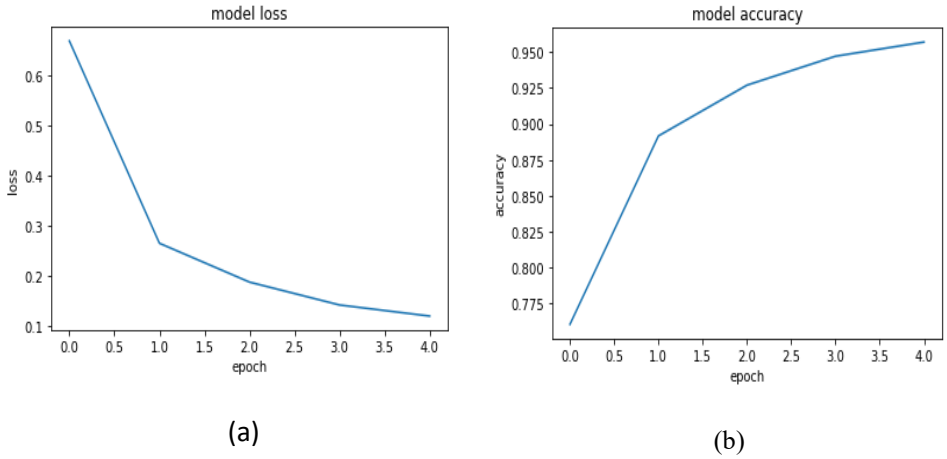
All these layers are compiled using 'adam' optimizer [24] and binary cross entropy is used as another training criterion. The model is trained using 5 epochs with batch size of 32. Once the training is being completed, a testing dataset is used for obtaining the final prediction results. The following table 2 shows the description of the model.

Table 2. Model Description

Layer No.	Layer Name	Layer Type	Output Shape	# Parameters
1	embedding	(Embedding)	(None,40,2)	20000
2	conv1d	(Conv1D)	(None, 38, 32)	224
3	bidirectional	(Bidirectional LSTM)	(None, 38, 128)	49664
4	bidirectional_1	(Bidirectional LSTM)	(None, 38, 64)	41216
5	Flatten	(Flatten)	(None, 64)	0
6	dense	(Dense)	(None, 64)	4160
7	dense_1	(Dense)	(None, 1)	65

4. RESULTS AND DISCUSSION

This section provides the training procedure results. The accuracy and loss obtained during each epoch is shown in Figure 2. As shown in Figure 2, as the number of epochs is increasing, the accuracy increases and the loss decreases. Finally, an accuracy of 0.9571 and loss of 0.1189 is reached by our proposed model during the last epoch of the training process. Table 3 shows the exact proportion of loss and accuracy for each epoch. Once the training procedure is completed, the test dataset is fitted to the model. In other words, the testing accuracy and error rate is measured at the end of 5^{th} epoch. Table 4 shows the testing accuracy and loss acquired by the model. As discussed, the training results exhibited by the C-RNN model shows good performance as it reaches accuracy almost close to 1.0. This training outcome is justified by the testing results which show a good generalization output.



(a) (b)

Figure 2. (a) Loss and (b) Accuracy obtained for each epoch during training.

Table 3. Exact Ratio of Training Time Loss and Accuracy

Epoch Number	#Samples	Time Taken	Loss	Training Set Accuracy
1	25000	53s	0.6684	0.7604
2	25000	51s	0.2643	0.8917
3	25000	52s	0.1867	0.927
4	25000	54s	0.141	0.9427
5	25000	54s	0.1189	0.9571

Table 4. Error Rate and Testing Accuracy of proposed method

Proposed Method	Loss	Accuracy
C-RNN	0.5158	84.73%

As compared to the existing research studies, our presented method indicates superior efficiency while making an automated sarcasm detection model. An existing research work carried out in [10] has reached the highest efficiency of 75.89%. An accuracy of 68.7% is achieved by [8] for accomplishing the sarcasm detection task. Highest efficiency of 69.13% in terms of Macro-F is exhibited by [15]. While considering the current work, this study has achieved an accuracy of 84.37% for sarcastic news identification. The uniqueness of this work lies in obtaining a fusion model that accommodates two dissimilar deep neural network layers. This work differs from its peer research works in terms of automated classification without incorporating the manual feature engineering task. The news contents are analysed extensively by the presented hybrid model that can detect the sarcastic clues by itself without any manual intervention. However, the pre-processing tasks are preceded by the presented classification technique for achieving the better result. The superiority of the classification technique also depends on choosing the right hyper-parameters during the implementation. Considering all these necessary operations, the proposed C-RNN approach can be exemplified as a computer aided system for decision making process for sarcasm detection field.

5. CONCLUSION

Automated Sarcasm detection is an interesting field because it self-comprehends the differences between sarcastic and lying patterns which would not be feasible by manual recognition process. Detecting sarcasm in social media enables capturing insight into the trend of current public opinion. This paper approaches an automated process that will discover unseen sarcastic sentiment on news twitter posts. Use of neural network is approached in this study in order to simulate human

brain-like operations. So, DL based implementation is favoured for this sarcasm detection domain which is indeed a complex event to be identified. This paper carries out a combined method that assembles convolutional layer as well as Bi-LSTM layer into an entity for recognizing hidden sarcastic patterns in tweet. This combined model is adjusted using necessary parameter tuning. Fine-tuning these parameters will assist in obtaining the best performance. The proposed model is capable of identifying sarcastic tweets with an accuracy of 84.73%. In conclusion, a computerized sarcasm detection system is implemented in this paper that is proficient to infer sarcasm from large databases with promising accuracy and optimized error rate.

REFERENCES

[1] M. Bouazizi and T. Otsuki, "A Pattern-Based Approach for Sarcasm Detection on Twitter," *IEEE Access*, vol. 4, pp. 5477–5488, 2016, doi: 10.1109/ACCESS.2016.2594194.

[2] R. Giora, O. Fein, J. Ganzi, N. A. Levi, and H. Sabah, "On negation as mitigation: The case of negative irony," *Discourse Process.*, vol. 39, no. 1, pp. 81–100, 2005, doi: 10.1207/s15326950dp3901_3.

[3] S. L. Ivanko and P. M. Pexman, "Context Incongruity and Irony Processing Context Incongruity and Irony Processing," no. 918551878, 2010, doi: 10.1207/S15326950DP3503.

[4] J. Schmidhuber, "Deep Learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015, doi: 10.1016/j.neunet.2014.09.003.

[5] M. Tom, "Recurrent neural network-based language model ´ s Mikolov Introduction Comparison and model combination Future work," pp. 1–24, 2010.

[6] H. C. Shin *et al.*, "Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning," *IEEE Trans. Med. Imaging*, vol. 35, no. 5, pp. 1285–1298, 2016, doi: 10.1109/TMI.2016.2528162.

[7] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF Models for Sequence Tagging," 2015.

[8] S. Lukin and M. Walker, "Really? Well. Apparently Bootstrapping Improves the Performance of Sarcasm and Nastiness Classifiers for Online Dialogue," vol. 1, 2017.

[9] S. Amir, B. C. Wallace, H. Lyu, P. Carvalho, and M. J. Silva, "Modelling context with user embeddings for sarcasm detection in social media," *CoNLL 2016 - 20th SIGNLL Conf. Comput. Nat. Lang. Learn. Proc.*, pp. 167–177, 2016, doi: 10.18653/v1/k16-1017.

[10] R. González-ibáñez and N. Wacholder, "Identifying Sarcasm in Twitter : A Closer Look," no. 2010, pp. 581–586, 2011.

[11] C. Chang and C. Lin, "LIBSVM : A Library for Support Vector Machines," vol. 2, no. 3, 2011, doi: 10.1145/1961189.1961199.

[12] R. E. Wright, "Logistic regression.," in *Reading and understanding multivariate statistics.*, Washington, DC, US: American Psychological Association, 1995, pp. 217–244.

[13] F. Barbieri, H. Saggion, and F. Ronzano, "Modelling Sarcasm in Twitter , a Novel Approach," pp. 50–58, 2014.

[14] Quinlan J.R, "Simplifying Decision Trees," *International Journal of Man-Machine Studies*, vol. 27, no. 3. pp. 221–234, 1987.

[15] Wang, Z and Y. R. Zelin Wang, Zhijian Wu, Ruimin Wang, "Twitter Sarcasm Detection Exploiting a Context-Based Model," *Lect. Notes Comput. Sci.*, pp. 77–91, 2015.

[16] H. M. Wallach, "Topic Modeling : Beyond Bag-of-Words," no. 1, pp. 977–984, 2006.

[17] A. Joshi, V. Sharma, and P. Bhattacharyya, “Harnessing context incongruity for sarcasm detection,” *ACL-IJCNLP 2015 - 53rd Annu. Meet. Assoc. Comput. Linguist. 7th Int. Jt. Conf. Nat. Lang. Process. Asian Fed. Nat. Lang. Process. Proc. Conf.*, vol. 2, no. 2003, pp. 757–762, 2015, doi: 10.3115/v1/p15-2124.

[18] A. Ghosh and D. T. Veale, “Fracking Sarcasm using Neural Network,” pp. 161–169, 2016, doi: 10.18653/v1/w16-0425.

[19] W. Liu, Y. Wen, Z. Yu, and M. Yang, “Large-Margin Softmax Loss for Convolutional Neural Networks,” 2016.

[20] Rishabh Misra (October,2018), "News Headlines Dataset For Sarcasm Detection" Version 2. Retrieved on 24.05.2020 available from https://www.kaggle.com/rmisra/news-headlines-dataset-for-sarcasm-detection.

[21] Y. Li and Y. Yuan, “Convergence analysis of two-layer neural networks with RELU activation,” *Adv. Neural Inf. Process. Syst.*, vol. 2017-Decem, no. Nips, pp. 598–608, 2017.

[22] M. R. Zadeh, S. Amin, D. Khalili, and V. P. Singh, “Daily Outflow Prediction by Multi Layer Perceptron with Logistic Sigmoid and Tangent Sigmoid Activation Functions,” *Water Resour. Manag.*, vol. 24, no. 11, pp. 2673–2688, 2010, doi: 10.1007/s11269-009-9573-4.

[23] “Keras.” [Online]. Available: https://keras.io.

[24] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–15, 2015.