Hindawi
Complexity
Volume 2019, Article ID 8641074, 12 pages
https://doi.org/10.1155/2019/8641074

Research Article

A New Type of Eye Movement Model Based on Recurrent Neural Networks for Simulating the Gaze Behavior of Human Reading

Xiaoming Wang,[1] **Xinbo Zhao**,[1] **and Jinchang Ren**[2]

[1]*National Engineering Laboratory for Integrated Aero-Space-Ground-Ocean Big Data Application Technology, School of Computer Science and Engineering, Northwestern Polytechnical University, Xi'an 710072, China*
[2]*Department of Electronic and Electrical Engineering, University of Strathclyde, Glasgow G1 1XW, UK*

Correspondence should be addressed to Xinbo Zhao; xbozhao@nwpu.edu.cn

Received 6 November 2018; Revised 11 February 2019; Accepted 27 February 2019; Published 24 March 2019

Guest Editor: Jungong Han

Traditional eye movement models are based on psychological assumptions and empirical data that are not able to simulate eye movement on previously unseen text data. To address this problem, a new type of eye movement model is presented and tested in this paper. In contrast to conventional psychology-based eye movement models, ours is based on a recurrent neural network (RNN) to generate a gaze point prediction sequence, by using the combination of convolutional neural networks (CNN), bidirectional long short-term memory networks (LSTM), and conditional random fields (CRF). The model uses the eye movement data of a reader reading some texts as training data to predict the eye movements of the same reader reading a previously unseen text. A theoretical analysis of the model is presented to show its excellent convergence performance. Experimental results are then presented to demonstrate that the proposed model can achieve similar prediction accuracy while requiring fewer features than current machine learning models.

1. Introduction

Using computers to simulate humans or to reproduce certain intelligent behaviors related to human vision is a typical computer vision task [1], such as simulating eye movements in reading. However, reading is complex cognitive behavior and the underlying cognitive process occurs only in the brain [2]. Modeling such behavior requires obtaining some explicit indicators via such methods as eye tracking.

When reading a text, the eyes of a skilled reader do not move continuously over the lines of text. Instead, reading proceeds by alternating between fixations and rapid eye movements called *saccades* [3]. This behavior is determined by the physiological structure of the human retina. Most of the optic nerve cells are concentrated in the fovea and only when the visual image falls in this area can it be "seen" clearly. Unfortunately, the fovea only provides about a 5-degree field of view [4]. Therefore, the reader needs to change the fixation point through successive saccades so that the next content falls on the fovea region of the retina. By analyzing eye movements during reading, we can quantify the reader's actions and model for reading. Eye tracking helps researchers to determine where and how many times subjects focus on a certain word, along with their eye movement sequences from one word to another [5]. Figure 1 shows an example eye movement trajectory from an adult reader.

Models of eye movement control have been studied in cognitive psychology [6–9]. Researchers integrated a large amount of experimental data and proposed a variety of eye movement models such as easy-rider (E-Z) reader [10] and saccade generation with inhibition by foveal targets (SWIFT) [11]. Although these eye movement models typically have parameters that are fit to empirical data, their predictions are rarely tested on unseen data [12]. Moreover, their predictions are usually averaged over a group of readers, while eye movement patterns vary significantly between individuals [13]. Predicting the actual eye movements that an individual will make while reading a new text is arguably a challenging problem.

Some recent work has studied eye movement patterns from a machine learning perspective [14–17]. These studies were inspired by recent work in natural language processing

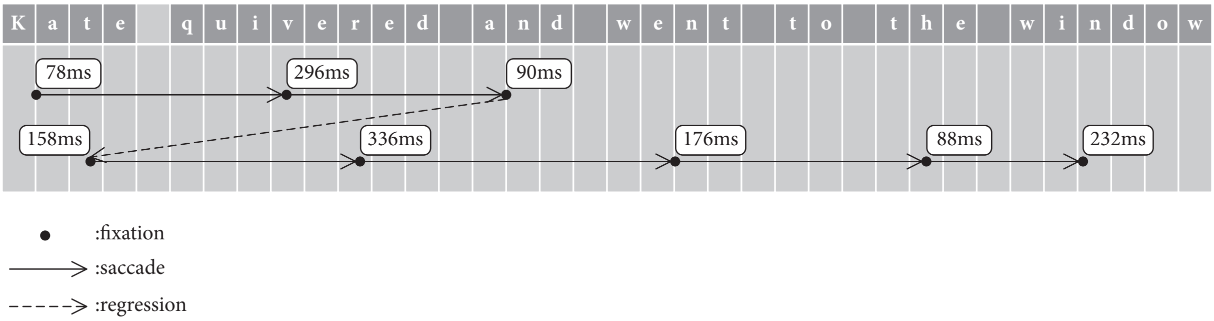

Figure 1: Eye track of an adult reader. Black dots indicate the position of the gaze point, the number next to each dot indicates the duration of each gaze point (in ms), and the arrow indicates the direction of the saccade.

(NLP) and are less tied to psychophysiological assumptions about the mechanisms that drive eye movements. The work presented in [14] was the first to apply machine learning methods to simulate human eye movements. The authors used a transformation-based model to predict word-based fixation of unseen text. Reference [17] applied a conditional random field (CRF) model to predict which words in a text are fixated by a reader. However, traditional supervised learning requires more features and preprocessing of data, which may lead to high latency in human-computer interaction applications.

Aided by their parallel distributed processing paradigm, neural networks have been widely used in pattern recognition and language processing because of their parallel distribution [18]. In 2010, Jiang [19] studied how to apply neural networks to multiple fields and provided a review of the key literature on the development of neural networks in computer-aided diagnosis. In 2011, Ren [20] proposed an improved neural network classifier that introduced balanced learning and optimization decisions, enabling efficient learning from unbalanced samples. In 2012, Ren [21] proposed a new balanced learning strategy with optimal decision making that enables effective learning from unbalanced samples and is further used to evaluate the performance of neural networks and support vector machines (SVMs).

In recent years, deep neural networks (DNN) have become a popular topic in the field of machine learning. DNN has successfully improved the recognition rate and some excellent optimization algorithms and frameworks have been proposed and applied. Guo (2018) proposed a novel robust and general vector quantization (VQ) framework to enhance both robustness and generalization of VQ approaches [22]. Liu (2018) presented an efficient bidirectional Gated Recurrent Unit (GRU) network to explore the feasibility and the potential of mid-air dynamic gesture based user identification [23]. Liu (2019) presented an end-to-end multiscale deep encoder (convolution) network, which took both the reconstruction of image pixels' intensities and the recovery of multiscale edge details into consideration under the same framework [24]. Wu (2018) proposed an unsupervised deep hashing framework and adopted an efficient alternating approach to optimize the objective function [25]. Wu (2019) proposed a Self-Supervised Deep Multimodal Hashing (SSDMH) method and demonstrated the superiority of SSDMH over state-of-the-art cross-media hashing approaches [26]. Luan (2018) developed a new type of deep convolutional neural networks (DCNNs) to reinforce the robustness of learned features against the orientation and scale changes [27]. Besides, some methods based on recurrent networks have been proposed, developed, and studied for natural language processing [28–30].

In this paper, we formalize the problem of simulating the gaze behavior of human reading as a word-based sequence labeling task (which is a classic NLP application). In the proposed method, the eye movement data of a reader reading some texts is used as training data and a bidirectional Long Short-Term Memory-Conditional Random Field (bi-LSTM-CRF) neural network architecture is used to predict the eye movement of the same reader reading a previously unseen text. The model is focused on achieving similar prediction accuracy while requiring fewer features than existing methods. However, it is worth emphasizing that in this study we focus only on models of where the eyes move during reading, and we will not be concerned with the temporal aspect of how long the eyes remain stationary at fixated words.

The remainder of this paper is organized into the following sections. Section 2 introduces the problem formulation. Section 3 proves the convergence of the model. Section 4 describes the layers of our neural network architecture. Section 5 discusses the training procedure and parameter estimation. Section 6 demonstrates the superiority of the proposed method with verification experiments. Section 7 concludes the paper with final remarks. Before ending the current section, it is worth pointing out the main contributions of the paper as follows.

(i) This paper proposes and tests a new type of eye movement model based on recurrent neural networks, which is quite different from previous research on eye movement model.

(ii) The convergence of the RNN-based model for predicting eye movement of human reading is proved in this paper.

(iii) An experiment of foveated rendering further demonstrates the novelty and effectiveness of recurrent neural networks for solving the problem of simulating the gaze behavior of human reading.

2. Problem Formulation

Experimental findings in eye movement and reading research suggest that eye movements in reading are both goal-directed and discrete [6]. This means that the saccadic system selects visual targets on a nonrandom basis and that saccades are directed towards particular words rather than being sent a particular distance. Under this view, there are a number of candidate words during any fixation, with each having a certain probability of being selected as the target for the subsequent saccade. For our purposes we will assume that a probabilistic saccade model assigns a probability to fixation sequences resulting from saccade generation over the words in a text. Let us use the following simple representations of a text and fixation sequence.

Let $\boldsymbol{R}$ denote a set of readers, and text $\boldsymbol{T}$ represent a sequence of word tokens (text) $(\mathrm{w}_1, \ldots, \mathrm{w}_\mathrm{n})$. Let $\boldsymbol{F}$ denote a sequence of fixation token positions in $\boldsymbol{T}$, generated by a reader $\boldsymbol{r}$ ($\boldsymbol{r} \in \boldsymbol{R}$). The fixation token positions set $\boldsymbol{S}(\boldsymbol{F})$ is the set of token positions that removes repetitive elements from $\boldsymbol{F}$, where $\boldsymbol{S} = \{\mathrm{S}_1, \ldots, \mathrm{S}_\mathrm{m}\} (1 \leq \mathrm{S}_\mathrm{i} \leq \mathrm{n})$.

$$S(F) = \arg\max p(S \mid T, r) \tag{1}$$

$p(\mathrm{S} \mid \mathrm{T}, \mathrm{r})$ is a reader-specific distribution of eye movement patterns given a text $\boldsymbol{T}$. For example, the text "*Kate quivered and went to the window*" is represented by $\boldsymbol{T}$ = (Kate, quivered, and, went, to, the, window). A sequence of fixations in a reader's reading of the text which is *Kate-quivered-and-went-to-the-window* is represented by $\boldsymbol{F} = (1, 2, 3, 1, 2, 4, 6, 7)$; and the corresponding fixation token positions set is $\boldsymbol{S}(\boldsymbol{F}) = \{1, 2, 3, 4, 6, 7\}$.

The next object of study is the prediction of the fixation point sequence $\boldsymbol{F}$ based on a specific reading event $\boldsymbol{E}$ involving the reader $\boldsymbol{R}$ reading the text $\boldsymbol{T}$. The training data consist of words and a Boolean value indicating whether they are fixation words, in the form of $((\mathrm{w}_1, \mathrm{boolFixation}_1), \ldots, (\mathrm{w}_\mathrm{n}, \mathrm{boolFixation}_\mathrm{n}))$.

$\boldsymbol{M}$ is a recurrent neural networks model. Given some text as input, the purpose of this model is to generate a sequence of fixation points that approximate human reading behavior. We evaluate the performance of model $\boldsymbol{M}$ by comparing the predicted fixation sequence set $\boldsymbol{S}_{\boldsymbol{M}}$ with the actual fixation sequence set $\boldsymbol{S}_{\boldsymbol{O}}$ observed in a reading experiment involving $\boldsymbol{R}$ and $\boldsymbol{T}$. To do this, we train $\boldsymbol{M}$ on a novel set of texts $\boldsymbol{X} = \{\mathrm{X}_1, \ldots, \mathrm{X}_\mathrm{m}\}$ generated by a reader $\boldsymbol{r} \in \boldsymbol{R}$. The goal is to infer

$$s^* = \arg\max_{s \in S(F)} p(S \mid M, X, r) \tag{2}$$

3. Convergence Analysis

For neural networks with $\boldsymbol{m}$ hidden nodes and $\boldsymbol{n}$ training data samples, if the ReLU activation function is used, a previous study in [31] has shown that, as long as $\boldsymbol{m}$ is sufficiently enough, the randomly initialized gradient descent algorithm converges to the global optimal solution. In this section, by following the gradient descent provably optimized method, the convergence performance of our RNN-based model is presented below.

First, we consider a recurrent neural network of the following form:

$$z^{(t)} = Ux^{(t)} + Wh^{(t-1)} + b \tag{3}$$

where $\boldsymbol{x}$ is the input vector, $\boldsymbol{h}$ is a hidden layer, $\boldsymbol{z}$ is an inactive hidden layer, $\boldsymbol{U}$ is the matrix of $\boldsymbol{x}^{(\mathrm{t})}$ to $\boldsymbol{h}^{(\mathrm{t})}$, $\boldsymbol{W}$ is the matrix of $\boldsymbol{h}^{(\mathrm{t}-1)}$ to $\boldsymbol{h}^{(\mathrm{t})}$, and $\boldsymbol{b}$ is the bias of the hidden layer.

Use the tanh activation function to activate all nodes of the layer $\boldsymbol{z}$ to the layer $\boldsymbol{h}$:

$$h^{(t)} = \phi\left(z^{(t)}\right) = \tanh\left(z^{(t)}\right) \tag{4}$$

Map the layer $\boldsymbol{h}^{(\mathrm{t})}$ to the layer $\boldsymbol{o}^{(\mathrm{t})}$:

$$o^{(t)} = Vh^{(t)} + c \tag{5}$$

where $\boldsymbol{V}$ is the matrix of $\boldsymbol{h}^{(\mathrm{t})}$ to $\boldsymbol{o}^{(\mathrm{t})}$. Use $\boldsymbol{o}^{(\mathrm{t})}$ to derive the predicted value $\hat{y}$ and loss function of the model.

$$\hat{y}^{(t)} = \sigma\left(o^{(t)}\right) = crf\left(\mathrm{o}^{(t)}\right) \tag{6}$$

$$L = -\sum_{t=1}^{n}\sum_{k=1}^{C} y_k^{(t)} \ln\left(\hat{y}_k^{(t)}\right) \tag{7}$$

where $\boldsymbol{crf}$ is an activation function, $\boldsymbol{T}$ is the transform matrix of $\boldsymbol{y}_{\mathrm{t}-1}$ to $\boldsymbol{y}_\mathrm{t}$, $\boldsymbol{S}$ is the state matrix of $\boldsymbol{y}_\mathrm{t}$, Z is a scaling factor, and λ and μ are weights, which is defined as

$$crf\left(o\left(t\right)\right) = \frac{1}{Z\left(o^{(t)}\right)} \exp\left(\lambda T_{y_{t-1}, y_t} o^{(t)} + \mu S_{y_t} o^{(t)}\right) \tag{8}$$

There are 3 parameters that need to be optimized, namely, U, V, and W. Among them, the optimization process of the two parameters of W and U needs to trace the historical data, the parameter V is relatively simple and only needs the current data, and then we will solve the partial derivative of the parameter V first.

$$\frac{\partial L}{\partial V} = \sum_{t=1}^{n} \frac{\partial L^{(t)}}{\partial o^{(t)}} \cdot \frac{\partial o^{(t)}}{\partial V^{(t)}} \tag{9}$$

The solution of the partial and partial derivatives of W and U is relatively complicated because of the need to involve historical data. Let us assume that there are only three moments, and then the partial derivative of L to W at the third moment is

$$\begin{aligned} \frac{\partial L^{(3)}}{\partial W} = {} & \frac{\partial L^{(3)}}{\partial o^{(3)}} \frac{\partial o^{(3)}}{\partial h^{(3)}} \frac{\partial h^{(3)}}{\partial W} + \frac{\partial L^{(3)}}{\partial o^{(3)}} \frac{\partial o^{(3)}}{\partial h^{(3)}} \frac{\partial h^{(3)}}{\partial h^{(2)}} \frac{\partial h^{(2)}}{\partial W} \\ & + \frac{\partial L^{(3)}}{\partial o^{(3)}} \frac{\partial o^{(3)}}{\partial h^{(3)}} \frac{\partial h^{(3)}}{\partial h^{(2)}} \frac{\partial h^{(2)}}{\partial h^{(1)}} \frac{\partial h^{(1)}}{\partial W} \end{aligned} \tag{10}$$

It can be observed that, at some point, the partial derivative of L to W needs to trace back all the information before this moment. We can write the general formula of L to the partial derivative of W at time t according to formula (10):

$$\frac{\partial L^{(t)}}{\partial W} = \sum_{k=0}^{t} \frac{\partial L^{(t)}}{\partial o^{(t)}} \frac{\partial o^{(t)}}{\partial h^{(t)}} \left(\prod_{j=k+1}^{t} \frac{\partial h^{(j)}}{\partial h^{(j-1)}} \right) \frac{\partial h^{(k)}}{\partial W} \tag{11}$$

$$\prod_{j=k+1}^{t} \frac{\partial h^{(j)}}{\partial h^{(j-1)}} = \prod_{j=k+1}^{t} crf' \cdot W_s \tag{12}$$

To prove that the RNN-based model will converge, it is only necessary to prove that there is an upper limit on the number of incorrect training examples on the training data set. The following lemmas are presented before the proof.

Lemma 1. *If the training data set is linearly separable, then there is a shortest distance from all training data points to the distance separating the hyperplanes, denoted as γ. Prove that $\|\hat{y}^{(k)} - y\|_2^2 \le (1 - \eta\gamma/2)^k \|\hat{y}^{(0)} - y\|_2^2$ is established.*

Proof. According to Cauchy inequality $\widehat{w} \cdot \widehat{w}_{opt} \le \|\widehat{w}\| \cdot \|\widehat{w}_{opt}\|$.

$$\begin{aligned} &\because \widehat{w}_k = \widehat{w}_{k-1} + \widehat{x}_t y_t \\ &\therefore \widehat{w}_k \cdot \widehat{w}_{opt} = \left(\widehat{w}_{k-1} + \widehat{x}_t y_t\right) \widehat{w}_{opt} \ge \widehat{w}_{k-1} \widehat{w}_{opt} + \eta\gamma \\ &\qquad \ge \widehat{w}_{k-2} \widehat{w}_{opt} + 2\eta\gamma \ge \cdots \ge k\gamma \\ &\therefore \left\|\widehat{y}^{(k)} - y\right\|_2^2 \le (1 - \eta\gamma/2)^k \left\|\widehat{y}^{(0)} - y\right\|_2^2 \end{aligned} \tag{13}$$

The proof is thus completed. □

Lemma 2. *Let $R = 4\sqrt{n}\|y - \widehat{y}^{(0)}\|_2 / \sqrt{m}\lambda_0$, then the number of misclassifications k of the algorithm on the training data set satisfies the inequality $\|y - \widehat{y}^{(k+1)}\|_2^2 \le (1 - kR/2)\|y - \widehat{y}^{(k)}\|_2^2$.*

Proof. At the k-th iteration, we have $\|\widehat{y}^{(k)} - y\|_2^2 \le (1 - \eta\gamma/2)^k \|\widehat{y}^{(0)} - y\|_2^2$. If k'=0,. . .,k, then we have for every $\boldsymbol{r} \in [m]$

$$\left\| w_r^{(k+1)} - w_r^{(0)} \right\|_2 \le \frac{4\sqrt{n}\left\|y - \widehat{y}^{(0)}\right\|_2}{\sqrt{m}\lambda_0} = R \tag{14}$$

Next, we calculate the difference in prediction between two consecutive iterations.

$$\begin{aligned} \widehat{y}^{(k+1)} - \widehat{y}^{(k)} &= \frac{1}{\sqrt{m}} \sum_{r=1}^{m} a_r (\sigma\left({w_r^{(k+1)}}^T x_i \right) \\ &\quad - \sigma\left({w_r^{(k)}}^T x_i \right) \end{aligned} \tag{15}$$

As in the classical analysis of gradient descent, we also need to bound the quadratic term.

$$\begin{aligned} \left\|\widehat{y}^{(k+1)} - \widehat{y}^{(k)}\right\|_2^2 &\le \eta^2 \frac{1}{m} \left(\sum_{r=1}^{m} \left\| \frac{\partial L\left(W^{(k)}\right)}{\partial w_r^{(k)}} \right\|_2 \right)^2 \\ &\le \eta^2 n^2 \left\|y - \widehat{y}^{(k)}\right\|_2^2 \end{aligned} \tag{16}$$

With these estimates, we are ready to prove the theorem.

$$\begin{aligned} \left\|y - \widehat{y}^{(k+1)}\right\|_2^2 &= \left\|y - \widehat{y}^{(k)} - \left(\widehat{y}^{(k+1)} - \widehat{y}^{(k)}\right)\right\|_2^2 \\ &\le \left(1 - \eta\lambda_0 + 2\eta n^2 R + 2\eta n^{3/2} R + \eta^2 n^2\right) \left\|y - \widehat{y}^{(k)}\right\|_2^2 \\ &\le \left(1 - \frac{kR}{2}\right) \left\|y - \widehat{y}^{(k)}\right\|_2^2 \end{aligned} \tag{17}$$

□

This indicates that there is an upper limit on the number of misclassifications, and the algorithm will converge when the final number of misclassifications reaches the upper limit. So if the data is linearly separable, then the model does converge.

4. Network Architecture

In this section, we describe a CNN-LSTM-CRF-based network architecture for reading eye movement prediction, consisting of a word embedding layer, a CNN layer, a bidirectional LSTM layer, and a CRF layer from bottom to top.

4.1. Word Embedding and CNN Layer.

It was shown in [28] that word embedding plays a crucial role in improving the performance of sequence labeling. We vectorize the text corpus by converting each text into a sequence of integers (each integer is an index of the mark in the dictionary), where the coefficients of each mark can be based on the number of words.

Previous studies, e.g., [32, 33], showed that convolutional neural networks (CNNs) are effective methods for extracting word form information from characters in a word and encoding it into a neural representation. Our CNN is similar to that used in [33], except that we use not only word embedding as CNN input, but also the linguistic features of the word.

4.2. RNN Layer.

One of the key features of recurrent neural networks (RNNs) is that they can be used to connect past information to current tasks, such as inferring the meaning of a current word from previous words. However, a RNN cannot connect past information when the gap between the relevant information and the current position increases.

Long Short-Term Memory (LSTM) network is a special RNN that can learn long-term dependencies through specialized design. A LSTM is also a multilayered type of the neural network, in which the single neural network layer contains four interactive sublayers, as shown in Figure 2.

In many sequence labeling tasks, it is best to have access to past (left) and future (right) contexts, while LSTM's hidden state does not get information from the future. An excellent solution is the bidirectional LSTM (bi-LSTM) [34], whose validity has been proven in previous studies. Therefore, we can effectively use past features (through the forward state) and future features (through the backward state). We use

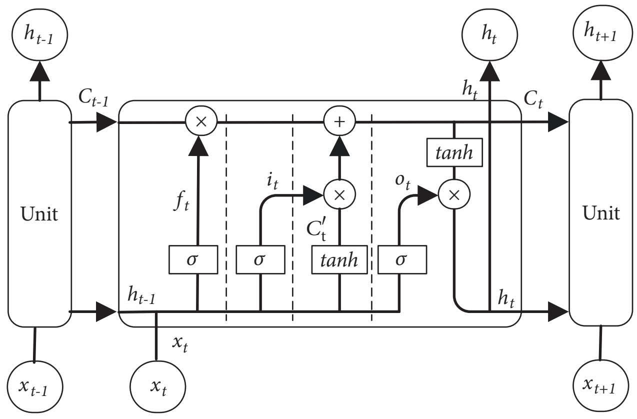

FIGURE 2: The repeating unit in LSTM networks.

backpropagation through time (BPTT) [35] to train bi-LSTM networks, which help to process multiple sentences at the same time.

4.3. CRF Layer. A conditional Random Fields (CRF) is a conditional probability distribution model whose nodes can be divided exactly into two disjoint sets X and Y, which are the observed and output variables, respectively. Under CRF, the inference problem for CRF is essentially the same as for a Markova Random Field (MRF) where an input random variable X is given, and the output Y is observed. The output variable Y represents a saccade target (fixation) label sequence, and the input variable X represents the word sequence that needs to be marked or labeled. The reading saccade target prediction problem to be solved is transformed into a sequence labeling problem in this paper. Under the condition that the random variable X is x, the conditional probability for the random variable Y to be valued as y is

$$P(y \mid x) = \frac{1}{Z(x)} \cdot \exp\left[\sum_{i,k} \lambda_k t_k (y_{i-1}, y_i, x, i) + \sum_{i,l} \mu_l s_l (y_i, x, i)\right] \quad (18)$$

In this expression, Z (x) is a scaling factor; t_k is a transfer feature function that depends on the current landing position and the previous position. s_l is a state feature function that also depends on the current landing position. The value of the feature function t_k and s_l is 1 or 0, which means when it meets the feature condition, the value is 1; otherwise it is 0. λ_k and μ_l are weights, which are obtained by training the model. Parameter training is achieved based on a maximum likelihood criterion and maximum a posteriori criterion, whose goal is to maximize the probability of correctly marking the target sequence in the training set.

Whether the current word (x) is a fixation word (y_i) depends not only on the feature value of the current word (x), but also on whether the previous word is a fixation word (y_{i-1}). This coincides with the characteristics of the linear chain CRF.

4.4. CNN-LSTM-CRF Architecture. In the final stage, we built our neural network model by feeding the output vector of the bi-LSTM into the CRF layer. Figure 3 details our network architecture.

5. Model Training

In this section, we provide detailed information on training neural networks. We use the *keras-contrib* library [36] to implement a neural network that contains useful extensions to the official *Keras* package [37]. Model training is run on the GeForce GTX 1070 graphical processing unit. It takes approximately 20 min to complete the model training using the setup discussed in this section.

5.1. Datasets. There are several eye-tracking corpora in existence. Among these, the Dundee corpus [38] is notable. However, the Dundee corpus is not publicly available due to licensing restrictions. Our experiments used data from the Provo corpus [39], which is publicly accessible and may be downloaded from the Open Science Framework at https://osf.io/sjefs. The eye-tracking corpus has eye movement data from 84 native English-speaking participants, all of whom read the complete 55 texts for comprehension, including online news articles, popular science magazine articles, and public domain works of fiction. Eye movements were documented using a SR Research EyeLink 1000 Plus eye-tracker with a spatial resolution of 0.01° and a sampling rate of 1000 Hz (see [39] for details).

For the experiments reported here, the corpus was randomly divided into three data sets in the following proportions: 60% texts for training, 20% texts for development and validation, and the last 20% texts for testing.

5.2. Features. Evidence from the psychological literature indicates that the selection mechanism of fixation and saccade is determined by the combination of low-level visual cues (such as word length) and language cognitive factors of the text [7, 40]. The linguistic factors known to influence whether or not to skip words are word length, frequency, and predictability; that is, shorter words in the text are easier to skip than longer words. Predictability involves high-level cognitive factors that are difficult to quantify. Thus, we use parts of speech (POS) as a proxy to represent the high-level cognitive factors. Words in the corpus are tagged for parts of speech using the Constituent Likelihood Automatic Word-Tagging System (CLAWS) [41]. Using the CLAWS tags, words are divided into nine separate classes. The passages contain a total of 682 nouns, 502 verbs, 364 determiners, 287 prepositions, 227 adjectives, 196 conjunctions, 169 adverbs, 109 pronouns, and 153 other words and symbols.

Ultimately, the features used by the neural network consist of a tokenized word, word length (for low-level visual features), and parts of speech (for high-level cognitive features).

5.3. Training Procedure. The model used in this paper was trained using general stochastic gradient descent (SGD),

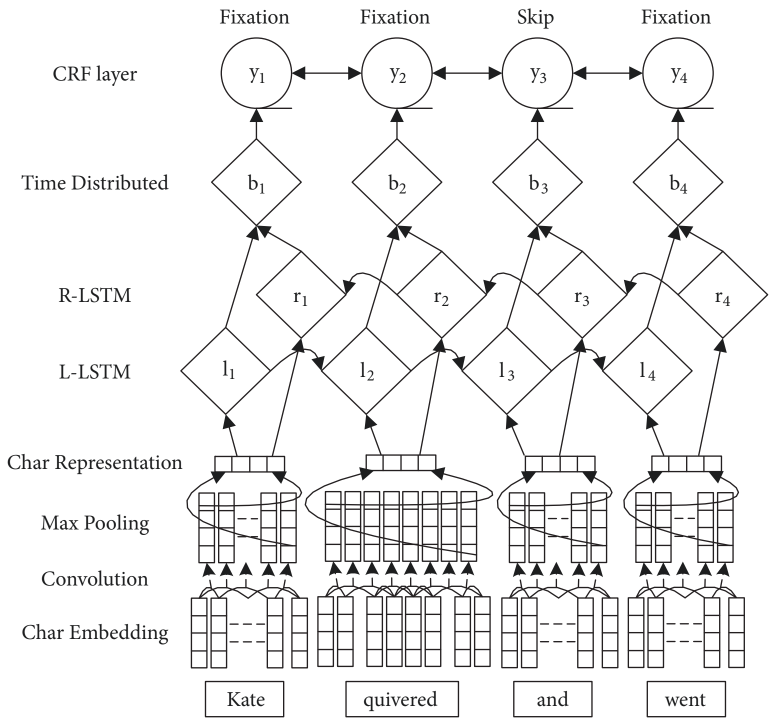

Figure 3: The CNN-LSTM-CRF architecture for predicting fixation in reading. The first layer is the embedding layer, which vectorizes the text corpus by converting each text into a sequence of integers. The second layer is the CNN layer, which extracts word form information from characters in a word and encoding it into a neural representation. The third layer is the bidirectional LSTM neural network, which can effectively use past (left) and future (right) contexts. The fourth layer is the CRF layer, which is used for sentence-level sequence labeling.

```
for epoch in epochs:
    for batch in batchs:
        (1) bi-LSTM model forward pass:
                forward pass for forward state LSTM
                forward pass for backward state LSTM
        (2) CRF layer forward and backward pass
        (3) bi-LSTM model backward pass:
                backward pass for forward state LSTM
                backward pass for backward state LSTM
        (4) update parameters
```

Algorithm 1: The model training procedure.

forward propagation, and backpropagation (BP) algorithms. The training procedure is shown in Algorithm 1.

For each training sample, the algorithm first initialized random weights and threshold parameters, then provided relevant input examples to the input layer neurons, and forwarded the signals layer by layer (input layer -> hidden layer -> output layer) until the output layer produced an output value. Then, the error of the output were calculated according to the output value, and then the error was reversely propagated to the neurons of the hidden layer, and finally the weight of the connection and the threshold of the neuron were adjusted according to the error calculated by the hidden layer neurons. The BP algorithm continually iteratively looped through the above steps until the conditions of stopping training were reached.

5.4. Tuning Hyperparameters. The hyperparameters that need to be determined include (1) word embeddings dimension, (2) word embeddings length, (3) convolution kernel size, (4) maxpool size, (5) neuron activation function type, (6) cost function, (7) weight initialization method, (8) number of neurons in each hidden layer, (9) optimizer, (10) learning rate, (11) learning epoch, and (12) batch size. Among these hyperparameters, (1) and (2) were determined by the size of the development set, (6), (7), and (8) were determined by some common strategies, and (3), (4), (5), (9), (10), (11), and (12) were determined by random search.

Since there were 1200 different words in the development set, and each sentence contained less than 60 words, we set the word embeddings dimension to 1200 and the word embeddings length to 60. In the one-dimensional convolution operation, it was equivalent to the feature extraction of *n_gram* using neural network, so the optional parameters were 2, 3, and 4.

To find the optima values for hyperparameters, we used the following strategies: first, we determined the type of activation function, and then determined the type of cost function and the method of weight initialization. Secondly, according to the network topology, the number of neurons in each hidden layer in the neural network was determined. Then, for the remaining hyperparameters, a possible value

Table 1: Hyperparameter settings.

Layer	Hyper-parameter	Value	Parameter Range
Embedding	output dim	32	/
	input dim	1200	/
	input length	60	/
CNN	kernel size	3	$[2, 3, 4]$
	maxpool size	3	$[2, 3, 4]$
	stride size	1	/
L-LSTM	units	32	/
R-LSTM	units	32	/
Time Distributed	units	50	/
Global	activation	ReLU	[Sigmoid, Tanh, ReLU]
	cost	cross_entropy	/
	kernel initializer	random_normal	/
	optimizer	SGD	[SGD, AdaDelta, Adam, RMSProp]
	learning rate	2.5	$[0.025, 0.25, 2.5]$
	learning epoch	100	$[50, 100, 150, 200]$
	batch size	16	$[8, 16, 32]$
	validation split	0.2	/

was randomly given first. In the cost function, the existence of the regular term was not considered first, and the learning rate was adjusted to obtain a suitable threshold. Half of the threshold was taken as the initial value in the process of adjusting the learning rate. The size of the batch was determined through experiments. Then we used the determined learning rate to carefully adjust the learning rate and the validation data to select good regularization parameters. After the regularization parameters were determined, we went back and reoptimize the learning rate. The number of epochs was determined by a whole observation through the above experiments.

Among all the parameters, the type of neuron activation function should be selected first. The Sigmoid, Tanh, and ReLU functions are the most commonly used activation functions [42], but the Sigmoid and Tanh functions will encounter the problem of gradient disappearance, which is not conducive to the extension of the neural network to a deeper structure. The ReLU overcomes this problem because it solves the problem of gradient disappearance and therefore allows the neural network to extend to deeper layers. So we chose the ReLU as the activation function here.

Since our task was a classification task, the cost function that should be used in the experiment was the cross-entropy. For an input layer with n_{in} neurons, the initialization weight is a Gaussian random distribution with a mean of 0 and a standard deviation of $1/\sqrt{n_{in}}$. We determined the optimizer by random search. The super-parameters to be tuned included SGD [42], AdaDelta [43], Adam [44], and RMSProp [45].

The adjustment steps of the learning rate were as follows: first, we choose the estimation that the cost on the training data immediately began to decrease rather than oscillate or increase as the learning rate threshold, and it was not necessary to be too precise to determine the magnitude. If the cost began to decline in the first few rounds of training, we gradually increased the magnitude of the learning rate. If the cost function curve began to oscillate or increase, we tried to reduce the magnitude until the cost fell at the beginning of the round. Taking half of the threshold determined the learning rate.

Table 1 summarizes the relevant hyperparameters used in the experiments determined by the development set, the common strategies, and the random search method. The last column in the table is the parameter range of the random search. In order to avoid the overfitting problem, we used a simple dropout strategy [46] to drop 10% of the redundant nodes.

6. Experimental Verification

6.1. Evaluation Metrics and Baselines. We followed the evaluation metrics and baselines in NN09 [12] and HMKA12 [17]. First, we quantified the number of words in the test set and used the fixation word rate for each subject as a baseline (see Table 2). Then, the accuracy of the fixation word prediction was used as an evaluation metrics, and the fixation/skip distribution of each word in the verification set was predicted. The accuracy of each predicted distribution was calculated from the distribution in the fixation data. Finally, the accuracy of each word in the validation set was averaged.

6.2. Result Analysis. Based on the analysis in Section 5.2, we set up features to predict the fixation points. The examined features can be classified into two types: low-level visual features and high-level cognitive features. In the experiments,

Table 2: Baseline rates for fixated words in the test data.

Subjects	Sub1	Sub2	Sub3	Sub4	Sub5	Sub6	Sub7	Sub8	Sub9	Sub10
# fixated words	1,907	2,158	2,120	1,788	1,666	2,040	1,856	1,989	1,537	2,046
Rate[%]	69.52	78.67	77.29	65.18	60.74	74.37	67.66	72.51	56.03	74.59
# words in test data	2,743(100%)									

Table 3: Mean accuracy and standard deviation (averaged over 100 runs) for the fixation prediction task.

Subjects	Baseline[%]	POS[%]		WL[%]		POS&WL[%]	
		Mean	Δ_{std}	Mean	Δ_{std}	Mean	Δ_{std}
Sub1	69.52	70.34	1.05	78.36	1.95	79.87	1.50
Sub2	78.67	78.91	1.10	81.39	1.53	83.87	1.80
Sub3	77.29	79.66	1.18	80.83	1.84	82.77	1.58
Sub4	65.18	68.85	1.82	76.65	2.03	78.52	2.04
Sub5	60.74	63.42	1.27	71.31	1.54	74.71	1.35
Sub6	74.37	75.21	1.12	77.63	1.50	79.29	1.05
Sub7	67.66	69.48	1.05	72.58	1.34	75.69	2.13
Sub8	72.51	73.75	1.85	76.67	2.09	78.47	1.76
Sub9	56.03	61.91	1.27	70.76	1.58	73.92	1.16
Sub10	74.59	75.79	0.96	79.09	1.40	79.91	1.72
Average	69.66	71.73	1.27	76.53	1.68	78.70	1.61

we explored the contribution of low-level visual and high-level cognitive features individually and in combinations to prediction accuracy.

Based on the experimental settings discussed in Section 5, we run at least 100 times with different random initialization of parameters. All experiments were trained using Stochastic Gradient Descent (SGD). We reported the mean accuracy and the standard deviation (Δ_{std}) and thus more fully characterize the distribution of accuracies in the predictions.

The experimental results in Table 3 show that the word length feature (for low-level visual features, denoted by "WL") has an accuracy of 76.53%, while the part of speech feature (for high-level cognitive features, denoted by "POS") provides less accuracy.

The achieved test performances can be plotted in a violin plot as shown in Figure 4. The violin plot is similar to a boxplot; however, it estimates from the samples the probability density function and depicts it along the Y-axis. If a violin plot is wide at a certain location, then achieving this test performance is especially likely. Besides the probability density it also shows the median as well as the quartiles. In Figure 4 we can observe that the WL feature usually results in a higher performance than the POS feature for the fixation prediction task. Hence, we can conclude that the low-level visual feature is a better option for this task.

We also considered combinations of the two feature types. From the Figure 5, we can see that adding other features to the WL feature barely contributes to an improvement in accuracy. Additionally, the influence of the POS feature on improving the accuracy is not obvious. The prediction accuracy obtained by the POS feature is similar to the baseline accuracy. These observations seem to imply that high-level cognitive features do not capture very much extra information when using the features separately. This would suggest that combined features work well only in conjunction with low-level visual features.

In summary, we can draw the conclusion that the low-level visual cues have a key impact on the selection of saccade targets compared with the high-level cognitive factors.

The authors in [12, 17] used the Dundee corpus to train and test their models. Owing to licensing restrictions, our experiments are based on the data from the Provo corpus. Because of the different experimental settings, we cannot simply compare our experimental results with those from [12, 17]. However, considering that we were able to obtain similar accuracy as those using NN09 [12] or HMKA12 [17], and, moreover, we used far fewer features and required much less preprocessing than did NN09 or HMKA12 (see Table 4), these results indicate that the proposed RNN-based model performs well in simulating reading eye movements.

6.3. Application Example. Interactive graphics environments require high refresh rates, high resolutions, and low latencies, each of which adds computational burden on the hardware. To address the problems, the state-of-the-art technology that integrates with eye tracking is known as *foveated rendering* [47]. Foveated rendering can make the fixation point that the user is focusing on clearer and replaces the adjacent area with a blurred image, which is in line with the mode of human vision. In this way, the machine does not have to render the entire picture in detail, which can greatly reduce the computational burden on the GPU. However, accurate fixation tracking systems are still expensive and can only be accessed by a limited number of researchers or companies [48]. Another way to compute the fixation point is to use an eye movement model that simulates the gaze behavior of human.

Table 4: Comparison between E-Z Reader, NN09, HMKA12, and RNN-based models.

Parameters	E-Z Reader	NN09	HMKA12	RNN-based Model
# training sentences	/	157.8	157.8	137.5
# training features	/	8	7	2
Average fixation accuracy	57.7%	69.5%	78.601%	78.702%

Table 5: Comparison between reference latency, HMKA12 latency, and RNN-based latency.

Count of pixels	Reference Latency		HMKA12 Latency		RNN-based Latency	
	Single-GPU	Multi-GPU	Single-GPU	Multi-GPU	Single-GPU	Multi-GPU
32M	780 ms	577 ms	96.2 ms	74.0 ms	27.6 ms	19.7 ms
16M	532 ms	410 ms	95.6 ms	65.4 ms	27.6 ms	19.8 ms
8M	385 ms	289 ms	95.0 ms	58.6 ms	27.4 ms	16.9 ms
4M	267 ms	212 ms	94.4 ms	62.2 ms	27.4 ms	17.4 ms
2M	98.6 ms	75.4 ms	93.8 ms	63.1 ms	27.1 ms	17.6 ms
1M	90.4 ms	72.1 ms	89.9 ms	63.7 ms	27.3 ms	18.3 ms

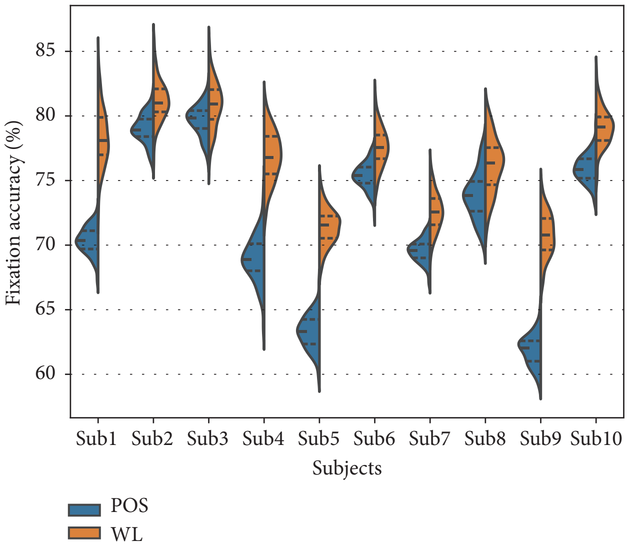

Figure 4: Performance on the fixation prediction task for various subjects using the POS feature or the WL feature.

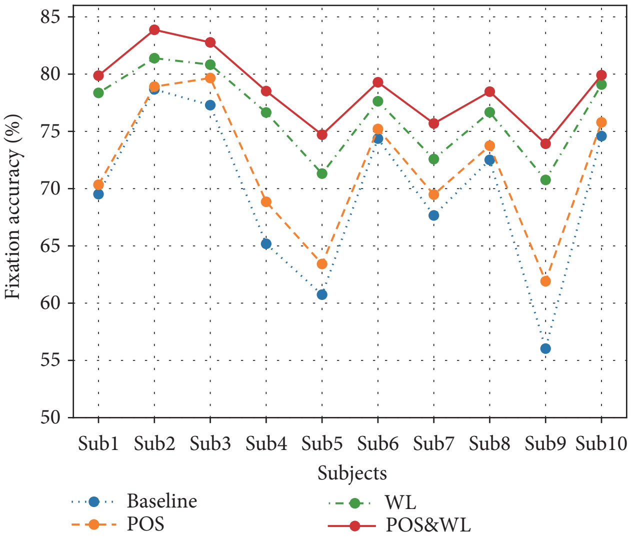

Figure 5: Fixation prediction accuracy comparison using different features on the validation data.

The proposed model requires a smaller number of input features than any of the alternatives, while the prediction accuracy is similar to that of current machine learning models. Requiring fewer features and reducing the preprocessing of data make the proposed model attractive in a range of human-computer application areas. For example, the latency can be reduced in an interactive graphics environment.

We constructed an application example using the architecture proposed in [49] to demonstrate that the proposed model can improve system performance by reducing latency. This was accomplished by a fixation-predicting architecture with a parallel client and server process that accesses a shared scene graph (Figure 6). Low latency and constant delay are ideal features of an interactive system. We examined the latency for single- and multi-GPU fixation-predicting implementations as well as for a standalone regular renderer for reference with a method first documented by Steed [50].

Table 5 lists the results obtained from the experiments and the results show the average delay obtained in several experiments. The reference render has a much higher latency, and the amount of delay depends on the number of pixels and frame rate in the scene. When the number of pixels is reduced to 4M or lower, the multi-GPU delay is slightly increased, which is affected by frequent asynchronous data transmission and context switching between threads, and a single GPU is not affected by this. In the case of small scenes, it is best to use direct rendering. However, when rendering is larger than 2M pixels, the fixation prediction method can significantly reduce the delay, and the RNN-based model has a lower delay.

6.4. Discussion. The RNN-based model achieves similar fixation prediction accuracy to current machine learning models

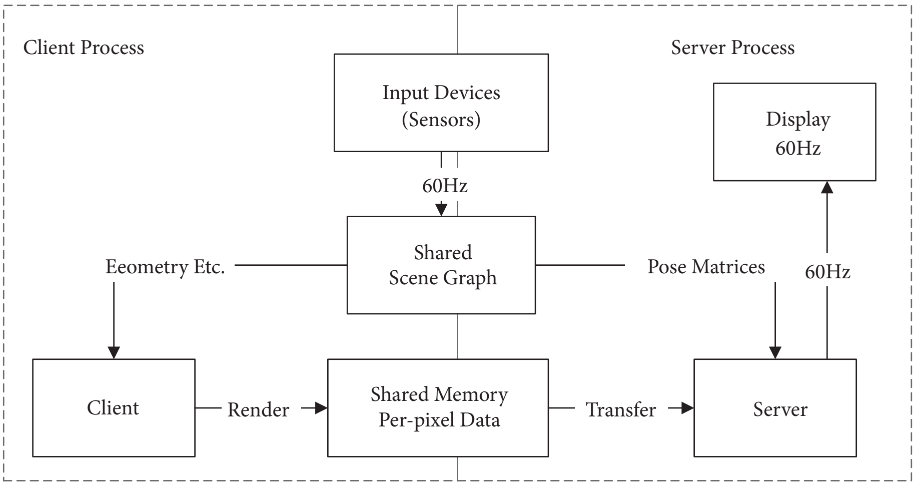

Figure 6: Overview of the gaze point rendering application architecture (adapted from [49]).

while requiring fewer features. We believe that there are two reasons for this. On the one hand, it uses a convolution operation and objectively increases the number of training samples. On the other hand, patterns of fixations and saccades are driven in part by low-level visual cues and high-level linguistic and cognitive processing of the text. CRF can account for the transfer features and state features of label sequences, in line with how humans handle low-level visual features. A RNN is a time-recursive neural network that can process and foresee events following particularly large intervals and delays in a time series, in correspondence with human handling of high-level visual features. Placing the RNN before the CRF is equal to utilizing the language relationships extracted by the RNN to train the CRF. The proposed model takes advantage of the context of the text sequence and the label sequence, which is more in line with the reality of the reading process.

7. Conclusions

In this paper, a new type of eye movement model was developed and evaluated in terms of its ability to simulate eye movements of human reading. Theoretical analysis demonstrated that the RNN-based model always converges to a global solution. In comparison with conventional eye movement models, the new approach was shown to achieve similar accuracy in predicting a user's fixation points during reading. In addition, the proposed model has less reliance on data features and requires less preprocessing than current machine learning models, which makes the proposed model attractive in a range of human-computer application areas. The verification results further demonstrate the novelty and efficacy of RNNs for simulating eye movements of human reading.

Data Availability

The eye-tracking corpus data used to support the findings of this study are available from the Open Science Framework at https://osf.io/sjefs.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grants nos. 61231016 and 61871326) and the General Project of Humanities and Social Sciences Research of Ministry of Education of China (Grant no. 18YJCZH180).

References

[1] G. Stockman and L. G. Shapiro, *Computer Vision*, Prentice Hall, 2001.

[2] X. J. Bai and G. L. Yan, *Psychology of Reading*, East China Normal University Press, 2017.

[3] H. X. Meng, *The Selection Mechanism of Saccade Target in Chinese Reading*, World Book Publishing Guangdong Co., Ltd., 2016.

[4] X. L. Liu, *Visual Neurophysiology*, People's Medical Publishing House, 2011.

[5] K. Rayner, "Eye movements in reading and information processing: 20 years of research," *Psychological Bulletin*, vol. 124, no. 3, pp. 372–422, 1998.

[6] R. Radach and G. W. Mcconkie, "Determinants of fixation positions in words during reading," in *Eye Guidance in Reading and Scene Perception*, G. Underwood, Ed., pp. 77–100, Elsevier, 1998.

[7] C. C. Jr, F. Ferreira, J. M. Henderson et al., "Eye movements in reading and information processing: Keith Rayner's 40 year legacy," *Journal of Memory and Language*, vol. 86, no. 1, pp. 1–19, 2016.

[8] S. G. Luke and K. Christianson, "Limits on lexical prediction during reading," *Cognitive Psychology*, vol. 88, no. 6, pp. 22–60, 2016.

[9] E. D. Reichle, "Computational models of reading: a primer," *Language and Linguistics Compass*, vol. 9, no. 7, pp. 271–284, 2015.

[10] E. D. Reichle, K. Rayner, and A. Pollatsek, "The E-Z reader model of eye-movement control in reading: comparisons to other models," *Behavioral and Brain Sciences*, vol. 26, no. 4, pp. 445–476, 2003.

[11] R. Engbert, A. Nuthmann, E. M. Richter et al., "SWIFT: a dynamical model of saccade generation during reading," *Psychological Review*, vol. 112, no. 4, pp. 777–813, 2005.

[12] M. Nilsson and J. Nivre, "Learning where to look: modeling eye movements in reading," in *Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL)*, pp. 93–101, Boulder, Colo, USA, 2009.

[13] N. Landwehr, S. Arzt, T. Scheffer, and R. Kliegl, "A model of individual differences in gaze control during reading," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1810–1815, Doha, Qatar, 2014.

[14] M. Hahn and F. Keller, "Modeling human reading with neural attention," 2016, http://cn.arxiv.org/abs/1608.05604.

[15] M. Nilsson and J. Nivre, "Towards a data-driven model of eye movement control in reading," in *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, pp. 63–71, Uppsala, Sweden, 2010.

[16] F. Matties and A. Søgaard, "With blinkers on: robust prediction of eye movements across readers," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 803–807, Seattle, Wash, USA, 2013.

[17] T. Hara, D. Mochihashi, Y. Kano et al., "Predicting word fixations in text with a CRF model for capturing general reading strategies among readers," in *Proceedings of the First Workshop on Eye-tracking and Natural Language Processing*, pp. 55–70, Mumbai, India, 2012.

[18] U. R. Acharya, S. L. Oh, Y. Hagiwara et al., "Deep convolutional neural network for the automated detection and diagnosis of seizure using EEG signals," *Computers in Biology and Medicine*, vol. 100, pp. 270–278, 2018.

[19] J. Jiang, P. Trundle, and J. Ren, "Medical image analysis with artificial neural networks," *Computerized Medical Imaging and Graphics*, vol. 34, no. 8, pp. 617–631, 2010.

[20] J. Ren, "ANN vs. SVM: which one performs better in classification of MCCs in mammogram imaging," *Knowledge-Based Systems*, vol. 26, no. 2, pp. 144–153, 2012.

[21] J. Ren, D. Wang, and J. Jiang, "Effective recognition of MCCs in mammograms using an improved neural classifier," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 4, pp. 638–645, 2011.

[22] Y. Guo, G. Ding, and J. Han, "Robust quantization for general similarity search," *IEEE Transactions on Image Processing*, vol. 27, no. 2, pp. 949–963, 2018.

[23] H. Liu, L. Dai, S. Hou, J. Han, and H. Liu, "Are mid-air dynamic gestures applicable to user identification?" *Pattern Recognition Letters*, vol. 117, pp. 179–185, 2019.

[24] H. Liu, Z. Fu, J. Han, L. Shao, S. Hou, and Y. Chu, "Single image super-resolution using multi-scale deep encoder-decoder with phase congruency edge map guidance," *Information Sciences*, vol. 473, pp. 44–58, 2019.

[25] G. Wu, J. Han, Y. Guo et al., "Unsupervised deep video hashing via balanced code for large-scale video retrieval," *IEEE Transactions on Image Processing*, vol. 28, no. 4, pp. 1993–2007, 2019.

[26] G. Wu, J. Han, Z. Lin, G. Ding, B. Zhang, and Q. Ni, "Joint image-text hashing for fast large-scale cross-media retrieval using self-supervised deep learning," *IEEE Transactions on Industrial Electronics*, 2018.

[27] S. Luan, C. Chen, B. Zhang, J. Han, and J. Liu, "Gabor convolutional networks," *IEEE Transactions on Image Processing*, vol. 27, no. 9, pp. 4357–4366, 2018.

[28] R. Collobert, J. Weston, L. Bottou et al., "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.

[29] K. Liang, N. Qin, D. Huang, and Y. Fu, "Convolutional recurrent neural network for fault diagnosis of high-speed train bogie," *Complexity*, vol. 2018, Article ID 4501952, 13 pages, 2018.

[30] C. A. Martín, J. M. Torres, R. M. Aguilar, and S. Diaz, "Using deep learning to predict sentiments: case study in tourism," *Complexity*, vol. 2018, Article ID 7408431, 9 pages, 2018.

[31] S. S. Du, X. Zhai, B. Poczos et al., "Gradient descent provably optimizes over-parameterized neural networks," 2018, http://cn.arxiv.org/abs/1810.02054.

[32] C. D. Santos and B. Zadrozny, "Learning character-level representations for part-of-speech tagging," in *Proceedings of the 31st International Conference on Machine Learning*, pp. 1818–1826, Beijing, China, 2014.

[33] J. P. Chiu and E. Nichols, "Named entity recognition with bidirectional LSTM-CNNs," 2015, http://cn.arxiv.org/abs/1511.08308.

[34] C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. A. Smith, "Transition-based dependency parsing with stack long short-term memory," 2015, http://cn.arxiv.org/abs/1505.08075.

[35] M. Boden, "A guide to recurrent neural networks and back propagation," *The Dallas project*, 2002.

[36] Keras-contrib library, 2018, https://github.com/keras-team/keras-contrib.

[37] Keras python library, 2018, https://keras.io.

[38] A. Kennedy, R. Hill, and J. Pynte, "The Dundee corpus," in *Proceedings of the 12th European Conference on Eye Movement*, Dundee, Scotland, 2003.

[39] S. G. Luke and K. Christianson, "The provo corpus: a large eye-tracking corpus with predictability norms," *Behavior Research Methods*, vol. 50, no. 2, pp. 826–833, 2018.

[40] A. Kennedy, J. Pynte, W. S. Murray, and S.-A. Paul, "Frequency and predictability effects in the dundee corpus: an eye movement analysis," *The Quarterly Journal of Experimental Psychology*, vol. 66, no. 3, pp. 601–618, 2013.

[41] R. Garside and N. Smith, "A hybrid grammatical tagger: CLAWS4," in *Corpus Annotation: Linguistic Information from Computer Text Corpora*, R. Garside, G. N. Leech, and T. McEnery, Eds., pp. 102–121, Longman, London, UK, 1997.

[42] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," 2013, http://cn.arxiv.org/abs/1211.5063v2.

[43] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," 2012, http://cn.arxiv.org/abs/1212.5701.

[44] D. Kingma and J. Ba, "Adam: a method for stochastic optimization," 2014, https://arxiv.org/abs/1412.6980.

[45] Y. N. Dauphin, H. De Vries, and Y. Bengio, "Equilibrated adaptive learning rates for non-convex optimization," *Advances in Neural Information Processing Systems*, vol. 35, no. 3, pp. 1504–1512, 2015.

[46] N. Srivastava, G. Hinton, and A. Krizhevsky, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[47] M. Weier, T. Roth, A. Hinkenjann et al., "Predicting the gaze depth in head-mounted displays using multiple feature regression," in *Proceedings of the 10th ACM Symposium on Eye Tracking Research and Applications (ETRA)*, pp. 1–9, Warsaw, Poland, 2018.

[48] E. Arabadzhiyska, O. T. Tursun, K. Myszkowski et al., "Saccade landing position prediction for gaze-contingent rendering," *ACM Transactions on Graphics*, vol. 36, no. 4, pp. 1–12, 2017.

[49] F. Smit, R. van Liere, S. Beck et al., "A shared-scene-graph image-warping architecture for VR: low latency versus image quality," *Computers and Graphics*, vol. 34, no. 1, pp. 3–16, 2010.

[50] A. Steed, "A simple method for estimating the latency of interactive, real-time graphics simulations," in *Proceedings of the VRST ACM Symposium on Virtual Reality Software and Technology*, pp. 123–129, Bordeaux, France, 2008.