LSTM Recurrent Neural Networks for Signature Verification

Conrad Tiflin and Christian W. Omlin
Department of Computer Science
University of the Western Cape
7535 Bellville, Cape Town, South Africa
Phone: +(27)(21) 959-3010
Fax : +(27)(21) 959-3006
{ctiflin,comlin}@uwc.ac.za

Abstract—This paper investigates the application of Long Short-Term Memory (LSTM) Recurrent Neural Networks (RNNs) to the task of signature verification. Traditional RNNs are capable of modeling dynamical systems with hidden states; they have been successfully applied to domains ranging from financial forecasting to control and speech recognition. We have applied on-line signature time series data to traditional LSTM, LSTM with forget gates and LSTM with peephole connections algorithms which was developed by [4]. It can be clearly seen in this pattern classification problem that traditional LSTM RNNs outperform LSTMs with forget gates and peephole connections. The latter also outperform traditional RNNs which can’t seem to even learn this task due to the long-term dependency problem.

Index Terms—Pattern Recognition, LSTM Recurrent Neural Networks, Handwritten Signature Verification, Gates, CEC.

I. INTRODUCTION

A considerable amount of research has been carried out in the area of handwritten signature verification using both static and dynamic approaches. According to the literature, the most acceptable results that have been attained was that of on-line signature verification systems. The reason for this was that more information about the signer’s signature was available during the authentication phase.

Recurrent Neural Networks (RNN’s) are able to recognise temporally extended patterns and therefore are appropriate tools for modeling time-verifying systems such as financial markets, physical dynamical systems, speech signals, etc. Networks can be used to recognize pattern sequences (e.g. speech recognition) or they can be used for forecasting future patterns (e.g. stock market prediction).

An individual’s handwritten signature can be captured and represented as a time series, \(x_1(t), x_2(t), \ldots, x_n(t)\) where \(t = 1, \ldots, T\). A novel RNN architecture is applied to signature time series which is able to discriminate between genuine signatures and forgeries with a high degree of accuracy.

II. MODELING OF TIME SERIES WITH RNN’S

A. Recurrent Neural Networks

Recurrent Neural Networks (RNN’s) have connections fed back into the nodes that exist in the hidden layers (Elman networks) or nodes that are fed back from the output layer (Jordan networks) or every node is connected to every other node in the network (fully RNN’s). These feedback connections enable these networks to create a ‘memory’ of past events that occurred numerous time steps ago.

The RNN’s core learning algorithm is gradient descent based. For pattern classification purposes, only supervised or associative learning in considered. This involves a teacher signal being assigned to each pattern indicating which class it belongs to.

The aim of gradient descent learning is to find the best possible set of weights in the weight space that produce the minimum amount of error. So, learning in RNN’s is accomplished by optimizing an error function \(E\) over all sequences. \(E\) consists of target outputs \(t_k\) and actual outputs \(a_k\).

\[
E(t) = 1/2 \sum (t_k(t) - a_k(t))^2
\]

The error at a time \(t\) is calculated for a particular pattern, so \(E(t)\) will represent the sum of all the errors over all the patterns residing in the dataset. The weights are then updated according to the following rule:

\[
\Delta w = -\eta E(t)
\]
The vanishing gradient effect

During the LSTM forward pass phase, the activations of the units are stored and during the backward pass these forward activations are recursively utilized and the errors also stored. The greater the number of time steps \( t \) that the network utilizes, the greater the memory required for storage. According to \([4, 5, 6, 7]\) BPTT and RTRL algorithms fail to learn patterns that extend greater than 10 time steps. Furthermore, \([1]\) points out that a given task will exhibit long-term dependencies if computation of the desired output at time step \( t \) depends on the input presented at an earlier time step. For a complete explanation of the problem consult \([1]\).

III. \textbf{Long-Short Term Memory Recurrent Neural Networks}

Traditional RNN’s are appropriate tools for modeling short sequences; however training is unlikely to converge when sequences have long-term dependencies, i.e. current states influence states that occur far in the future.

The Long-Short Term Memory (LSTM) recurrent network architecture has been designed specifically for modeling time series with long-term dependencies.

LSTM RNN’s are similar in structure to fully RNN’s with the exception that the hidden layer units consist of memory blocks \([5, 6, 7]\). The purpose of these memory blocks is to minimize the amount of network parameters by sharing the same multiplicative gating units via memory cells. These gating units control input and output to every single memory cell residing in a block.

At the center of the memory cell is a self-recurrent linear unit known as the Constant Error Carousel (CEC). The self recurrent weighted connection is fixed at 1.0 . The CEC produces an activation which represents the state of the cell at a given instant. The CEC’s main purpose is to prevent the backpropagated error from diminishing. The gates in a cell are there to control activation flow during the forward pass and activation flow during the backward pass. This is key in preventing noise and extraneous input from entering the memory cell.

One problem that is evident with traditional LSTM RNN’s is that continuous time series that are not preprocessed i.e. segmented into subsequences, will cause the cell state to grow ad infinitum. The solution to this is to replace CEC’s weight with a forget gate. Forget gates core purpose is to reset the current memory block once there contents have expired. According to

1) The net cell input is calculated as:

\[
z_{c_j}(t) = \sum_m w_{c_jm} y_m(t - 1)
\]

\[
y_i n_j(t) = f_i n_j(z_{i_j}(t))
\]

\[
z_i n_j(t) = \sum_m w_{in_jm} y_m(t - 1)
\]

2) The memory block forget gate activation:

\[
y_f(t) = f_f(z_{\varphi_j}(t)), z_{\varphi_j}(t) = \sum_m w_{\varphi_jm} y_m(t - 1)
\]

3) The new cell state:

\[
s_{c_j}(t) = y_{\varphi_j}(t) s_{c_j}(t - 1) + y_i n_j y g(z_{c_j}(t))
\]

\[
s_{c_j}(0) = 0.
\]

4) The cell output:

\[
y_{c_j}(t) = y_{out_j}(t) s_{c_j}(t).
\]

5) The output gate activation:

\[
y_{out_j}(t) = f_{out_j}(z_{out_j}(t))
\]

\[
z_{out_j}(t) = \sum_m w_{out_jm} y_m(t - 1).
\]

6) Finally the activation of the output units:

\[
y_k(t) = f_k(z_k(t)), z_k(t) = \sum_m w_k y_m(t).
\]

\( f \) represents a sigmoid function of range \([0, 1]\).

\[
f(x) = \frac{1}{1 - e^{-x}}
\]

It is important to note that forget gates are not necessary in the pattern classification task of signature verification since an individuals signature has a definite beginning and end and is not continuous.

![Fig. 2. a LSTM RNN with one memory block and cell](image-url)
For certain tasks, network performance can be harmed if all the gates are not allowed to inspect the CEC when the output gate is closed. A weighted peephole connection is introduced which solves this problem.

The backward pass for LSTM RNNs is a blend of backpropagation and a truncated version of RTRL. Consult [6] for full derivation of the learning algorithm. Once the error signal arrives at the output layer, it gets scaled by the output gate, it then enters the memory cell’s linear CEC where it can flow back indefinitely without ever being modified. This depends on the forget gate’s activation being close to 1.0. This is what allows arbitrary time steps between an input event and a target signal to be bridged and ultimately allows LSTM RNNs to work. Its important to note that memory blocks don’t exchange error signals.

IV. Signature Verification

Handwritten signature verification involves the classification process which ultimately strives to learn the manner in which an individual makes use of their muscular memory (hands, fingers and wrist) to reproduce a signature [8]. According to [8], given a population of signers there tends to exist a high degree of variability among them. In some cases the statics of a signature might be extremely complex while others appear to be simple, as if they could be forged quite easily.

An individual’s signature can be identified by two classes of features namely, static and dynamic. Static features are those that are concerned with the overall shape of the signature whereas signature dynamics represents the manner in which the signature is signed during the acquisition process. When developing a signature verification scheme, the main aim is to achieve the lowest possible Equal Error Rate (ERR). The ERR is simply the intersection between the False Rejection Rate (FRR) and the False Acceptance Rate (FAR). The FRR or type-I error represents the number of genuine signatures that the system rejects whereas the FAR or type-II error represents the number of fraudulent signatures that the system accepts. Since the relationship between FRR and FAR is inversely proportional, trying to decrease the one will inherently cause an increase in the other.

Its important to highlight that no two genuine signatures of an individual are exactly the same as they tend to vary in their statics and dynamics. As a result the process of verifying human signatures cannot be considered as a insignificant pattern recognition problem.

Signature Verification is essence a two-class classification problem (fig. 4). An individual’s signature can either be genuine or fraudulent.

A. Data Acquisition

An individual’s signature is acquired using a pressure sensitive tablet and stylus. The digitized signature consists of x and y pen tip coordinates, pen tip pressure and two polar angles.

- x and y coordinates represent the position of the signature in the 2-dimensional cartesian plane
- pen tip pressure indicates the amount of force that is applied to the tablet surface via the stylus
- polar angles indicates the manner in which the stylus is being held at a given instant.

These vector components can then be represented as a time series which can be statistically analysed and modeled using a LSTM RNN since a signature pattern can typically extend for time steps greater than 200.

B. LSTM RNN Model Setup

To determine the simplest type of model that will solve the problem, one needs to follow an iterative approach which will encompass the following:

- LSTM RNN Architecture:
  - select the memory block size
  - select the number of memory cells
  - select the output gain function

- LSTM RNN Connectivity Options:
  - forget gates
  - peephole connections

- LSTM RNN Learning Options:
  - type of learning algorithm
  - adjust the learning rate
  - adjust the momentum rate

- Train and Test the Network

During the backward pass phase the weights are increased by a minuscule amount. If these additions happen to have the same sign, the resulting magnitude of the weight tends to become unbounded. If very large weights are generated...
quickly ie. (greater than 15) the learning and momentum rates need to be adjusted smaller.

The function used for the output kernel was the logistic sigmoid with range \([0, 1]\). Various other kernel functions could be used to try and simplify learning for the network such as the 8020 RMSE function which represents the target 1 as 0.8 and the target 0 as 0.2.

C. Performance Results

The signature database that was utilized contained signatures of 51 individuals each consisting of genuine and fraudulent (over-the-shoulder, home-improved and professional) signatures. The data set was split into 70% signatures for training and the remaining 30% for testing.

The simplest LSTM architecture that could model an individual’s signature consisted of 5 memory blocks each of which contained 4 memory cells. A small learning rate of 0.0001 was set along with a high momentum rate of 0.9 to avoid network instability. An output gain of three was used. This is simply a multiplier before the hidden layer activations is passed through the output layer squashing function. The results for an individuals signature are as follows:

**Traditional LSTM RNN’s**

<table>
<thead>
<tr>
<th>Epoch Count</th>
<th>Mean Training Error</th>
<th>Mean Testing Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>7460</td>
<td>0.07699</td>
<td>0.09934</td>
</tr>
</tbody>
</table>

**LSTM RNN’s with Forget Gates**

<table>
<thead>
<tr>
<th>Epoch Count</th>
<th>Mean Training Error</th>
<th>Mean Testing Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>36913</td>
<td>0.08135</td>
<td>0.09979</td>
</tr>
</tbody>
</table>

**LSTM RNN’s with Forget Gates and Peephole Connections**

<table>
<thead>
<tr>
<th>Epoch Count</th>
<th>Mean Training Error</th>
<th>Mean Testing Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>19310</td>
<td>0.05286</td>
<td>0.093414</td>
</tr>
</tbody>
</table>

Fig. 5. Training and Testing Error Curve of Traditional LSTM RNN’s

Fig. 6. Training and Testing Error Curves for LSTM RNN’s with Forget Gates

Fig. 7. Training and Testing Error Curves for LSTM RNN’s with Peephole connections

devolved by [4]. It can be clearly seen in this pattern classification problem that traditional LSTM RNN’s outperform LSTMs with forget gates and peephole connections. The latter also outperform traditional RNN’s which can’t seem to even learn this task due to the long-term dependency problem.

ACKNOWLEDGMENT

This research has been funded through the Telkom-Cisco Centre of Excellence for IP and Internet Computing.

REFERENCES


**Biography**

Conrad Tiflin (ctiflin@uwc.ac.za) is a Telkom Masters student, finishing up his Masters Degree with the Intelligent Systems Group of the Department of Computer Science at the University of the Western Cape. His research focuses on the field of Biometrics and is sponsored by the Telkom Center of Excellence. His research interests span Recurrent Neural Networks (RNNs) and their applications to time series prediction. His M.Sc. thesis specifically addresses Signature Verification using Long Short-Term Memory RNNs.

Christian W. Omlin (comlin@uwc.ac.za) is Professor of Computer Science at the University of the Western Cape (UWC) and Head of the Intelligent Systems Research Group. His research interests include the application of machine learning methods to intelligent information retrieval and service delivery on the World Wide Web, knowledge-based neurocomputing and hybrid systems, integration of verbal and signed communication and biometrics.

He has published about 50 papers in bookchapters and international conference proceedings and journals and is the co-author of the upcoming book "Knowledge Representation and Acquisition in Recurrent Neural Networks: Foundations, Algorithms, and Applications" to be published by World Scientific Publishing Company. He also holds two U.S. patents on neural networks.

He is a member of IEEE, ACM, and INNS and regularly reviews manuscripts for international journals and conferences. He is Head of the Telkom-Cisco Centre of Excellence for IP and Internet Computing. He has served on the advisory boards of the African Institute for Mathematical Sciences and the International Conference on Application and Theory of Artificial Intelligence.