A Spiking Network that Learns to Extract Spike Signatures from Speech Signals

Amirhossein Tavanaei and Anthony S. Maida
Center For Advanced Computer Studies
University of Louisiana at Lafayette, LA 70504, USA
tavanaei@louisiana.edu, maida@cacs.louisiana.edu

Abstract

Spiking neural networks (SNNs) with adaptive synapses reflect core properties of biological neural networks. Speech recognition, as an application involving audio coding and dynamic learning, provides a good test problem to study SNN functionality. We present a novel and efficient method that learns to convert a speech signal into a spike train signature. The signature is distinguishable from signatures for other speech signals, representing different words, thereby enabling digit recognition and discrimination in devices that use only spiking neurons. The method uses a small SNN consisting of Izhikevich neurons equipped with spike timing dependent plasticity (STDP) and biologically realistic synapses. This approach introduces an efficient, fast, and multi-speaker strategy without error-feedback training, although it does require supervised training. The new simulation results produce discriminative spike train patterns for spoken digits in which highly correlated spike trains belong to the same category and low correlated patterns belong to different categories. The signatures can be used as a set of feature maps for classification. Our experiments compare two simple classifiers for spoken digit recognition in both clean and noisy environments.

Keywords: Spiking neural networks, STDP, speech recognition, neural model, spike signatures, speech signal coding.

1 Introduction

Spiking neural networks (SNNs) with adaptive synapses reflect core properties of nearly all biological networks. One of the most prolific mechanisms of synaptic modification in biological networks is known as spike-timing-dependent plasticity (STDP) [1]. Mechanisms of this type take into account the relative spike times of pre- and postsynaptic neural spikes to adjust the strength of a synapse connecting two neurons. The question of what STDP accomplishes in a learning framework is, and has been, under intense investigation. Spiking neurons and STDP learning rules have been applied in diverse fields of pattern recognition and classification [2,3,4,5,6,7] such as learning and information processing of visual features [8,9,10] and speech recognition [11]. Our work studies the performance of a novel STDP-trained SNN on isolated spoken word recognition. Our network produces, as output, discriminative spike signatures. These can be used to readily identify the class of an input pattern.

In the case of temporal coding, the main goal of the learning in these investigations is to match the output spike trains with a target spike train for a given category. Pfister et al. [12] developed a supervised learning algorithm to obtain a precise spike-time code using STDP and optimizing the likelihood of the postsynaptic firing at predefined, desired firing times. In the current study we generate a desired spike train (signature) for each output neuron using a supervised learning process. The desired firing times are obtained from input spike samples instead of one or more prespecified firing times. Therefore, the desired spike train reflects the uniqueness of the input characteristics and can enhance the pattern recognition accuracy of previous work. Each desired spike train type — a spike pattern signature — specifies a specific class in contrast to other classes.

Prior research has shown that spoken digit classification divides into two components: extracting features from the speech signal and then subsequent classification. There are many feature extraction methods. The Mel frequency cepstral coefficient (MFCC) feature extraction method combined with a hidden
Markov model (HMM) classifier is a popular method in automatic speech recognition (ASR) systems [13]. Although HMMs give excellent performance in ASR, they do not appear to offer a direct comparison to human brain mechanisms. To address this, much research has studied neurocomputational approaches to ASR mimicking the biological inspiration of the human auditory system [14,15,16]. The auditory system has components for encoding the raw signal (inner ear) and generating appropriate spike trains (cochlea). [17] has compared the MFCC approach with a cochlear model. In the present work, we use a simple approach based on Fibonacci coefficients.

For a digit-recognition SNN, converting the speech signal to a spike train is an important first step. Dibazar et al., for instance, proposed a feature extraction method using a dynamic synaptic neural network for isolated word recognition [18]. They achieved 99% accuracy, but at the expense of high computational complexity for estimating the network parameters. In another study, Loiselle et al. utilized the cochlear Gammatone filter bank and predefined thresholds to generate spike trains in French spoken word recognition based on rank order coding with an SNN [19]. They achieved an accuracy rate of about 65%.

Several studies have used reservoir-based approaches to spoken digit classification [17,20,21,22,23]. See [24] for a review of reservoir-based approaches in general. As a whole, this work has been quite successful in achieving near perfect digit recognition performance with robustness to noise. Reservoirs were proposed as a solution to the slow convergence of recurrent neural networks, which were of interest because of their ability to store temporal information. Reservoir computing avoids the convergence issue by using a suitably structured RNN as a reservoir (temporal memory) that is not trained. The reservoir can consist of either spiking [17] or non-spiking [20] neurons. Training is reserved for a linear readout layer that trains rapidly. The linear readout performs well because the RNN maps the inputs to a higher dimensional space in which the categories are more likely to be linearly separable.

Although the reservoir may or may not be built from spiking neurons, to our knowledge, in the context of speech recognition, the trainable readout layer has never used spiking neurons. In particular, the state of a spiking reservoir is low-pass filtered [17,20] before being sent to the readout layer. This allows the readout layer to be rapidly trained using any traditional non-spiking method for a single-layer architecture [23].

Although the reservoir approach described above yields outstanding performance on spoken digit recognition, the question of training a spiking network in the context of speech recognition remains unresolved. The present paper explores the training of a single layer spiking network for spoken digit recognition. Its novelty lies in the fact that it trains spiking neurons and then generates output spike signatures extended in time. The training of spiking neurons is entirely avoided in the reservoir-based approaches. Our approach is not incompatible with reservoir-based approaches. Our single trainable spiking layer can be used, in principle, as the readout layer to a reservoir. Although this may not yield performance improvement in the reservoir approach, since the performance is already so good, it can yield a more consistent architecture that only uses spiking neurons. Specifically, since spike signatures are the output, these outputs can be used as inputs for further processing that use architectures designed to accept spike trains as input [23,26].

In this study we train a small feedforward network of spiking neurons using a combination of supervised Hebbian and anti-Hebbian STDP. The combination of Hebbian and anti-Hebbian STDP is also novel. The trained network can be used to extract category-specific spike signatures. Also, the trained synaptic weights extract input signatures invariant to different speakers (male and female) and signal variations.

In summary, our method is novel in three ways: 1) we use a Fibonacci-based approach to feature extraction; 2) we train spiking neurons for spoken digit classification; and, 3) we use a combination of supervised Hebbian and anti-Hebbian STDP learning.

2 Feature Extraction

Feature extraction converts a raw signal into a more usable form. The speech signal is divided into small overlapping time sections called speech frames. The Hamming window, which is commonly used in discrete time signal processing, is used in signal framing due to its frequency features [27]. Our SNN needs a fixed number of frames, \( N \), for each spoken digit. In our experiments, \( N = 40 \). The length, \( L \), of a spoken digit can vary from 600 to 1,100 milliseconds. The frames have an overlap fraction, \( \gamma \). In our experiments, \( N = 40 \), \( L \) can vary from 600 to 1,100 milliseconds, the overlap is 50 percent, so \( \gamma = 0.5 \). The window size (frame length) in milliseconds is calculated based on \( L \), \( N \), and \( \gamma \), as shown below.
Figure 1: Spectrogram for the digit *seven*. The horizontal axis shows time (600-1100 ms word length) and vertical axis shows the frequency which increases from bottom (0 Hz) to top (4000 Hz). Color shows energy with yellow > red > blue.

\[
\text{window size}_{\text{(ms)}} = \frac{L_{\text{(ms)}}}{N(1 - \gamma) + \gamma} \tag{1}
\]

Because window size increases with signal duration, spoken words pronounced slowly have longer frames in comparison to words pronounced quickly.

After framing, a small feature vector for each frame is extracted. There are several methods for speech frame feature extraction such as MFCC and MFDWC [28]. We instead use a feature vector extracted from the frame’s frequency spectrum, as explained below.

### 2.1 Frequency spectrum

As a speech signal unfolds in time, the power of its frequency spectrum varies. This can be visualized in a spectrogram as shown in Fig. 1. Spectrograms can be used to identify spoken words phonetically, and to analyze the audio files in specific frames. Spectrum calculation of a frame is shown in Eq. (2). The spectrum values are calculated for all of the frames temporally to represent the speech signal spectrogram. Fig. 1 shows the spectrogram for the spoken digit *seven*.

\[
\text{Spectrum} = \log |\text{FFT(frame)}|^2 \tag{2}
\]

### 2.2 Frequency band

Low frequencies in the spectrogram have more energy and information relevant to classification than the high frequencies (cf. Fig. 1). Thus, an effective feature vector provides more resolution in low frequencies. This is obtained by using incrementally spaced frequency bands. We use a novel method for small problems to create the frequency bands from the Fibonacci sequence. This sequence provides good frequency band sizes for a small number of features.

A separate feature vector is calculated for each frame. A frame encompassing an \( R \) Hz frequency range can be divided into \( M \) frequency bands. In this paper, \( R = 4000 \) Hz and \( M = 5 \). If the first frequency band length is \( x \), then the filter bank containing \( M = 5 \) bands will have lengths of \( x, x, 2x, 3x, 5x \). Specifically, each frame represented in the \( R = 4000 \) Hz frequency range is divided into \( M = 5 \) bands with lengths of (333.3, 333.3, 666.7, 1000, and 1666.7) as shown in Fig. 2. \( x \) is chosen so that the equality below is satisfied.

\[
R = \sum_{i=1}^{5} \text{fib}(i) \cdot x = 12x \tag{3}
\]

The value of each element in a feature vector is the average energy over the range given in Fig. 2. For example, the first feature value codes the average energy in the range \( 0 - 333.3 \) Hz.
Figure 2: Five frequency bands for 0–4000 Hz frequency range.

Figure 3: Spike coding: RS neuron spikes with $I_{\text{inj}}$ equal to 150 (left) and 250 (right) for a duration of $T = 100$ ms.

3 Input Spike Generation

We use the Izhikevich model regular spiking (RS) neuron \(^{[29]}\) to convert a feature component to a spike train, as seen in Eqs. (4) through (6). Extracted features control the value of the injected input current, $I_{\text{inj}}$, to an afferent $y$ unit. The $I_{\text{inj}}$ drives the system. For the $y$ units, the only input is $I_{\text{inj}}$, so $I_{\text{inj}} = I_{\text{tot}}$ in the equations below. A larger total current causes more frequent spikes as seen in Fig. 3.

\[ C \frac{dV}{dt} = k(V - V_{\text{rest}})(V - V_{\text{th}}) - U + I_{\text{tot}} \] \hspace{1cm} (4)

\[ \frac{dU}{dt} = a[b(V - V_{\text{rest}}) - U] \] \hspace{1cm} (5)

and the reset equation

\[ \text{if } V > V_{\text{peak}} : V = c, \ U = U + d, \ \text{AP is emitted} \] \hspace{1cm} (6)

The spike time is the time step at which the membrane potential, $V$, becomes greater than $V_{\text{peak}}$. $U$ specifies a recovery factor inhibiting the AP and keeps the membrane potential near the resting value, $V_{\text{rest}}$. The neuron capacity ($C$), threshold ($V_{\text{th}}$), $V_{\text{peak}}$, and symbols $a, b, c, d$ are constants, specified in Table 1, whose values control the dynamic characteristics of the system and cause the neuron to have regular spiking behavior.

Each feature vector component drives one RS neuron ($y$ unit, as explained in the next section), causing it to generate a spike train over a fixed duration $T = 100$ milliseconds.

Table 1: RS neuron parameters for both $y$ and $z$ units.

| Parameter   | Value | Parameter | Value |
|-------------|-------|-----------|-------|
| $V_{\text{rest}}$ | -60   | $a$       | 0.03  |
| $V_{\text{th}}$  | -40   | $b$       | -2    |
| $V_{\text{peak}}$| 35    | $c$       | -50   |
| $C$           | 100   | $d$       | 100   |
| $K$           | 0.7   | $U_0$     | 0     |
| $\Delta T$    | 0.1   | $I_{\text{inj}}$ | variable |

4 Network Architecture

Our network is trained using STDP with labeled data. After training, the network can generate signature spike trains for the ten digit categories. The signature spike trains can be compared with spike trains from testing data to perform classification. The network architecture appears in Fig. 4. It consists of:
Figure 4: Network architecture. Each of the $N$ feature vectors produced by a feature extraction box has $M = 5$ components. Each component becomes $I_{inj}$ to a $y$ unit which then converts it to a spike train. The number of $y$ units equals $N \cdot M = 200$. Output units $z_1 - z_{10}$ represent the ten digit categories. The teacher signal controls whether Hebbian STDP or anti-Hebbian STDP is applied. $N \cdot M$ synapses project to each $z$ unit. There are $10 \cdot N \cdot M = 2000$ trainable synapses.

1. The network input consists of feature vector input from $N = 40$ frames. The $N$ frames cover the duration of the speech input stream. Each feature vector has $M = 5$ components as described in Sec. 2. For training, the sequential input is buffered and then presented to the network simultaneously. For testing, the procedure is slightly different.

2. The feature values are given to $y$ units that are implemented as RS neurons (configured according to parameters in Table 1). Each $y$ unit accepts one of five feature vector components which serves as its $I_{inj} = I_{tot}$ input value as described in Sec. 3. There are a total of $N \cdot M = 200$ $y$ units.

3. An output layer of ten $z$ units which correspond to the ten spoken digit categories (class labels). These are also implemented as RS neurons with the same parameter configuration as the $y$ units (Table 1). Their input consists entirely of synaptic input from the 200 $y$ units, namely $I_{syn} = I_{tot}$. The $y$ units are fully connected to the $z$ units. The $z$ units are trained according to the procedure described in Sec. 5.

4. Finally, there is a teacher that monitors the $z$ units in order to determine the form of the STDP used in training. If the target unit spikes at a given time step, it undergoes case 1 of Hebbian STDP and the rest of the (nontarget) units undergo case 1 of anti-Hebbian STDP. If the desired unit does not spike, it undergoes case 2 of Hebbian STDP and the rest of the units undergo case 2 of anti-Hebbian STDP. The teaching signal is only used for the training phase. In testing mode, a spike signature (explained in Sec. 6.3) is used for classification.

5 Learning

Learning is localized to the incoming synapses of the output neurons.

5.1 Neuron Model

Fig. 5 (left) shows the simulation circuit of a neuron in the model. The dashed box represents the spike generation step described in Eqs. (4 – 6). The branches marked $G_1$ to $G_3$ represent synaptic conductances for three synapses. If the neuron is a $y$ unit, then there is no synaptic input, only injected current $I_{inj}$. 

1
Figure 5: Left: Computation of $I_{\text{syn}}$ with three synaptic inputs, showing $I_{\text{syn}}$, $I_{\text{inj}}$, and $I_{\text{tot}}$. Right: Graph of synaptic conductance change over time after receiving a spike, $K_{\text{syn}}=1$ and $\tau=2$.

In this case, $I_{\text{tot}} = I_{\text{inj}}$. For a $z$ unit, there is synaptic input $I_{\text{syn}}$, but no injected current. In this case, $I_{\text{tot}} = I_{\text{syn}}$. Each $z$ unit has $N \cdot M = 200$ incoming synapses, corresponding to the afferent 200 $y$ units.

Learning occurs by modifying the synaptic conductances. $G_k(t)$ denotes the synaptic conductance change over time for synapse $k$ caused by receiving a single input spike to that synapse. The $\alpha$–function (Eq. 7) models the conductance time-course of the synapse. Fig. 5 (right) shows the $\alpha$–function graph for one synapse receiving one spike at time $t$.

$$G(t) = K_{\text{syn}} \cdot t \cdot e^{-t/\tau} \quad (7)$$

$K_{\text{syn}}$ controls the conductance amplitude. This is what is adjusted during learning. Synaptic weight adjustments change the value of $K_{\text{syn}}$ according to Eq. (11).

$\tau$ is the time at which the synapse reaches its maximum conductance. $t$ represents the elapsed time since the most recently received spike.

When multiple spikes are received in succession before a conductance drops to zero, the successive conductance effects are added linearly according to Eq. (8). Specifically, the total conductance of $N \cdot M$ input synapses with $N_{\text{rec},k}$ ($k = 1 : N \cdot M$) spikes is calculated by summing linearly over the synapses and input spikes:

$$G_{\text{tot}} = \sum_{k=1}^{N \cdot M} \sum_{j=1}^{N_{\text{rec},k}} K_{\text{syn},k}(t - t_{k,j}^f)e^{-(t-t_{k,j}^f)/\tau} \quad (8)$$

where $t_{k,j}^f$ is the spike time of spike $j$ for synapse $k$. $N_{\text{rec},k}$ denotes the number of spikes received by synapse $k$. The total synapse current $I_{\text{tot}}$ is given by:

$$I_{\text{syn}}(t) = \sum_{k=1}^{N \cdot M} E_{\text{syn},k}G_{\text{syn},k}^{\text{tot}}(t) - V(t) \sum_{k=1}^{N \cdot M} G_{\text{syn},k}^{\text{tot}}(t) \quad (9)$$

In our simulations, $E_{\text{syn},k} = 0$.

### 5.2 Spike Timing Dependent Plasticity (STDP)

Weight adjustment at a synapse is governed by the relative spike times of its pre- and postsynaptic neurons (Eq. [10]) in conjunction with the teacher feedback. The teacher feedback dictates the form of the STDP, whether it be Hebbian or anti-Hebbian. In the case of normal Hebbian STDP, if the postsynaptic spike is generated immediately after receiving the presynaptic spike, the presynaptic spike has a causal role in the output neuron firing. The synaptic weight is thus increased (LTP). Conversely, if a postsynaptic spike occurs before the presynaptic spike, the strength is reduced (LTD), as seen in the equation below.

$$\Delta w_{ji} = \begin{cases} 0.01A e^{-\frac{(t_{j}^f - t_{i}^r)}{\tau}} & t_{j}^f - t_{i}^r \geq 0, \ A > 0 \\ 0.01B e^{-\frac{(t_{j}^f - t_{i}^r)}{\tau}} & t_{j}^f - t_{i}^r < 0, \ B < 0 \end{cases} \quad (10)$$

In the above, the first case (Case 1) covers LTP and the second case (Case 2) covers LTD. Both cases are decaying exponentials that decay with the distance between and pre- and postsynaptic spikes. $A > 0$ and $B < 0$ scale the amplitude of the exponential, and $\tau+$ and $\tau-$ are the respective time constants.
Eq. (10) describes Hebbian STDP. To obtain anti-Hebbian STDP, we swap the cases. The teacher determines which \( z \) units undergo Hebbian versus anti-Hebbian STDP. During training, whenever a \( z \) unit emits a spike, it undergoes some form of STDP. If the \( z \) unit represents the target category, then it undergoes Hebbian STDP. Otherwise, it undergoes anti-Hebbian STDP.

In addition, the change in synaptic weights contributes to a change in the conductance amplitude, \( K_{\text{syn}} \), in the \( \alpha \)-function model.

We link the weight adjustment to the adjustment of \( K_{ji} \), which is used in the \( \alpha \)-function model (Eq. 7), by using the equation below.

\[
\Delta K_{ji} = \Delta w_{ji} K_{ji}
\]  

We now summarize the simulation’s operation during training. The simulation is advanced using \( \Delta t = 0.1 \) ms time steps using forward Euler (which is adequate for this particular problem). The \( y \) and \( z \) units are updated in a manner consistent with a feedforward sweep. Whenever a \( z \) unit fires, the teacher determines which variant of STDP to apply for that unit. We also renormalize the weights, using \( L_1 \), after each training epoch.

6 Experimental Methods and Results

6.1 Data Preparation

Our experiments were conducted using the Aurora dataset of spoken digits recorded from different male and female speakers [30]. The dataset was used for three purposes: training the network (500 samples), testing the network without noise (500 samples), and testing the network with noise (500 noisy samples, SNR=10 dB).

For training, 500 spoken digit samples, with 50 representatives for each digit (0 – 9), were randomly sampled from the Aurora dataset. Each digit sample was divided into 40 frames with 50% overlap Hamming windows, according to Eq. (1). The feature vector for a frame was obtained by applying the Fourier transform to the wave data and calculating the average energy of the five Fibonacci-scaled bands. This produces \( N = 40 \) feature vectors of \( M = 5 \) components. These are concatenated into a global feature vector of \( N \cdot M = 200 \) components. The feature vector values form the \( I_{\text{inj}} \) input to the \( y \) units.

6.2 Training

Before training the weights are initialized to uniform random values between 0 and 1 and then normalized using \( L_1 \) norm.

For each training instance, the network operates according to the following steps. The global feature vector is presented to the \( y \) units for a duration of \( T = 100 \) milliseconds. The \( y \) units generate spikes from their respective \( I_{\text{inj}} \) input as illustrated in Fig. [3]. Each of the ten output neurons receives 200 spike trains of duration 100 ms via 200 trainable synapses. The 2000 incoming synapses to the \( z \) unit layer were trained such that 200 synapses representing the digit category undergo Hebbian STDP and the remaining synapses undergo anti-Hebbian STDP.

In addition, the synaptic weights were renormalized after the presentation of each training. Because convergence was rapid, training was stopped after 100 epochs. Fig. [6] shows the trained synaptic weights arranged so that they can be compared with a spectrogram such as that shown in Fig. [1]. Each point \( (f, v) \) in this figure shows the synapse passing a spike train in respect to frame \( f \) and feature value \( v \) in ranges of 1 to 40 and 1 to 5 respectively.

6.3 Testing method

In testing mode the network generates spike signatures. Both target and test spike signatures are generated. The spike signatures are used to help visualize the network response.
6.3.1 Target spike signatures

We first explain how target spike signatures are generated. The target spike signatures are generated for each of the ten digit categories after training is complete. The trained network is used to generate these signatures. Before generating a target spike signature for a given category, we create a representative input feature set for that category. This involves resuming the training data, specifically taking an average over the training data for the category. Specifically, we averaged over the extracted feature coefficients for each digit class in the training set (50 samples per class) to obtain representative feature input for that class. That yielded $40 \times 5$ values for each new representative sample. These ten new items were given to the trained network to generate a target spike signature for each category.

We now explain how a representative feature set for a category, as calculated above, is used to generate a target spike signature. In spike signature mode, each input frame is processed individually and sequentially, rather than simultaneously. A frame, which contains 5 feature values, is converted to a spike train (with $T = 5$ ms and $\Delta T = 0.1$ ms) by passing through the corresponding inputs to the trained network. Each of the forty frames is passed through the network sequentially such that a spike signature of an idealized duration equal to 200 ms is obtained ($40 \times 5$ ms). The resulting target spike trains appear in Figs. 7 (left) and 8 (left) before and after training respectively.

6.4 Spike signature results

An example set of spoken digits 0 to 9 was selected to show the test spike signatures. These samples were not used in training. Testing spike signatures were generated analogously to target spike signatures, however, using a test sample as input. The resulting test spike trains after training appear in Fig. 8 (right). Fig. 7 shows spike train signatures before network training. The spikes are roughly uniformly distributed and dense. Comparison with Fig. 8 after training shows that meaningful spike signatures emerge. Comparison between spike signatures in Fig. 8 (left and right) show that signatures for test digits resemble the corresponding target signatures. Emitted spikes for the same digits show similar temporal patterns visually. For example, the target signature for digit six in Fig. 8 (left) is similar to the test signature for digit six (right). Specifically, its temporal patterns are similar where they have uniformly distributed spikes between 40 to 80 ms. However, the temporal patterns of the other digits (0-5, 7-9) have a large distance from the digit six target signature.

6.5 Classification performance results

To evaluate the network and feature maps (spike signatures), two classification methods were implemented. The first method was the maximum response method. This looked at the number of output spikes for each unit during the 200 ms test interval. In this method, information about the temporal coding of the output spikes was lost. Since all output intervals were of the same length, this is equivalent to using a rate coding scheme. The second method was the spike signature method. This method looked at the net input to an
Figure 7: Before training. Target spike trains based on 50 samples per class (left) and example test spike trains (right) for spoken digits 0 to 9. Each spike train has a duration of $T = 200$ ms.

Figure 8: After training. Target spike trains based on 50 samples per class (left) and example test spike trains (right) for spoken digits 0 to 9. Each spike train has a duration of $T = 200$ ms.
output unit grouped over each input frame. There were 40 input frames. This method preserved temporal
information at the resolution of 40 bins and 5 ms per bin. The net input corresponds to $I_{tot}$ in Fig. 5 in
which five input synapses are used (corresponding to the number of features in a frame). This method was
used in lieu of comparing spike signatures directly because it was easier to implement. The net input is
directly related to the probability of emitting a spike, so it was directly related to output spike signatures.
Comparison of these two methods gives an indication of the effectiveness of using a spike signature in
comparison to using the neural response rate code.

6.5.1 Maximum response method

In this method, the recognized digit was taken to be the one that emitted the most spikes over the 200
ms output interval. 500 unused samples from the Aurora data set were used for testing. Table 2 shows
the confusion matrix for isolated spoken digit recognition using this method. It shows the number of
occurrences for each event. The diagonals along the matrix indicate correct responses. Overall accuracy,
as defined by the percentage of correct responses, was 77%. The rightmost entry in each row of the table
shows the hit rate for each digit. That is defined as the percentage of time the network classified the
digit as, say “one”, when “one” was actually presented. The bottom entry in each column indicates the
misclassification rate. Additionally, to assess the performance of the network in the presence of noise, 500
noisy samples were used. The network was also evaluated by using noisy spoken digits (SNR=10 dB).
Table 3 shows the confusion matrix for noisy samples.

The overall performance of the network is given in Table 4 (top section). Specifically, the overall
accuracy rate is 77 percent for the clean test stimuli. The overall accuracy drops to 62 percent when stimuli
with noise is used. The performance of the network using maximum response classifier was low and
consistent with the performance expected from a single layer network.

However, the spike signatures shown in Fig. 8 indicate that they should be good discriminators. Thus,
in the next classifier we use information extracted from the network to capture spike signatures.

6.5.2 Spike signature method

The precursors to the spike signatures of the output units obtained from the trained network contain input
values that are directly related to spike events. These were taken as surrogates for the output of the network.
We then used these outputs to train an SVM classifier (think of it as a readout classifier).

The overall performance of the classifier is given in Table 4 (bottom section). Specifically, the overall
accuracy rate is 91 percent for the clean test stimuli. The overall accuracy drops to 70 percent when stimuli
with noise is used. Tables 5 and 6 show the confusion matrices for isolated spoken digit recognition in the
clean and noisy conditions, respectively.

Better performance of the second method (spike signature) determines the network ability in extracting
discriminative information from the input signals which can be fed into a classifier or next layer in a deep
belief network. Furthermore, extracting better information from the spike signatures would enhance the
classification correctness.

6.5.3 Discussion of results

The spike signature method at over 90 percent accuracy clearly outperformed the max response method
which achieved close to 77 percent accuracy. Performance dropped significantly when noise was added.
Other results are summarized in Table 4.

7 Conclusion

Our model represents a novel method for creating spike train signatures from spoken digits. It uses the
Izhikevich RS neuron model combined with STDP learning. The learning was implemented by Hebbian
and anti-Hebbian STDP controlled by a teaching signal.

The trained network produced target spike signatures for the ten spoken digits. Test samples that were
applied to the proposed SNN model, produced similar patterns for the same categories and different patterns
for the non-similar categories. The proposed SNN provided a fast pattern signature extraction system for
Table 2: Confusion matrix obtained using maximum response method for spoken digit recognition without noise. 500 unused samples were used for this test. Overall accuracy rate was 77%, calculated by summing along diagonal to count number or correct answers and dividing by 500. Off-diagonal rows entries indicate number of misses for that target. Off-diagonal columns entries indicate number of detection errors. Bottom right entry is diagonal total.

| Desired digit | Recognized | Row totals | Hit rate (%) |
|---------------|------------|------------|--------------|
|               | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 |   |     |
| 1             | 28 | 3 | 0 | 2 | 0 | 0 | 2 | 5 | 1 | 6 |   | 35 | 88.2|
| 2             | 0  | 46 | 2 | 0 | 0 | 2 | 0 | 2 | 0 | 2 |   | 54 | 85.2|
| 3             | 0  | 10 | 29 | 2 | 0 | 3 | 0 | 3 | 2 | 0 |   | 49 | 59.2|
| 4             | 1  | 1 | 0 | 41 | 2 | 0 | 0 | 1 | 0 | 3 |   | 51 | 80.4|
| 5             | 0  | 0 | 0 | 0 | 44 | 3 | 0 | 0 | 2 | 0 |   | 49 | 89.8|
| 6             | 0  | 7 | 1 | 0 | 0 | 37 | 1 | 3 | 0 | 0 |   | 49 | 75.3|
| 7             | 3  | 2 | 1 | 0 | 1 | 9 | 20 | 2 | 6 | 1 |   | 45 | 44.4|
| 8             | 0  | 3 | 6 | 2 | 1 | 3 | 0 | 42 | 0 | 0 |   | 57 | 73.7|
| 9             | 4  | 0 | 0 | 1 | 0 | 1 | 0 | 40 | 0 | 46 |   | 61 | 75.5|
| **Column totals** | **55** | **71** | **40** | **45** | **51** | **61** | **23** | **55** | **53** | **46** | **500** |

Table 3: Confusion matrix obtained using maximum response method for spoken digits with noise SNR=10 dB. 500 unused noisy samples were used for this test. Overall accuracy rate was 62%, calculated by summing along diagonal to count number or correct answers and dividing by 500. Off-diagonal rows entries indicate number of misses for that target. Off-diagonal columns entries indicate number of detection errors. Bottom right entry is diagonal total.

| Desired digit | Recognized | Row totals | Hit rate (%) |
|---------------|------------|------------|--------------|
|               | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 |   |     |
| 1             | 28 | 3 | 0 | 2 | 0 | 0 | 2 | 5 | 1 | 6 |   | 35 | 51.9|
| 2             | 0  | 38 | 8 | 4 | 0 | 6 | 0 | 0 | 1 | 57 |   | 66.7|
| 3             | 0  | 11 | 31 | 2 | 0 | 4 | 1 | 1 | 0 | 3 |   | 53 | 58.5|
| 4             | 0  | 4 | 1 | 41 | 0 | 3 | 2 | 0 | 0 | 0 |   | 51 | 80.4|
| 5             | 3  | 0 | 0 | 1 | 36 | 0 | 3 | 0 | 5 | 0 |   | 48 | 75.0|
| 6             | 0  | 10 | 0 | 0 | 31 | 4 | 0 | 0 | 0 | 45 |   | 68.9|
| 7             | 2  | 1 | 6 | 2 | 1 | 5 | 20 | 0 | 5 | 3 |   | 45 | 44.4|
| 8             | 0  | 3 | 4 | 3 | 1 | 6 | 0 | 0 | 1 | 34 |   | 44 | 40.9|
| 9             | 4  | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 35 | 5 |   | 50 | 70.0|
| **Column totals** | **37** | **80** | **54** | **59** | **39** | **58** | **45** | **20** | **56** | **52** | **500** |

Table 4: Overall performance accuracy

| Classifier  | Description | No noise | 10 dB noise |
|-------------|-------------|----------|-------------|
| Max response | Average hit rate (%) | 76.3 | 62.1       |
|             | Average misclassification rate (%) | 21.3 | 34.2 |
|             | Overall Accuracy | 76.6 | 62.3 |
| Spike signature | Average hit rate (%) | 90.9 | 70.9 |
|             | Average misclassification rate (%) | 9.3 | 29.9 |
|             | Overall Accuracy | 90.8 | 70.2 |
Table 5: Confusion matrix obtained using net input method for spoken digit recognition without noise. 500 unused samples were used for this test. Overall accuracy rate was 91%, calculated by summing along diagonal to count number or correct answers and dividing by 500. Off-diagonal rows entries indicate number of misses for that target. Off-diagonal columns entries indicate number of detection errors. Bottom right entry is diagonal total.

| Desired digit | Recognized |          |          |          |          |          |          |          |             |           |          |          |          | Hit rate (%) |
|---------------|------------|----------|----------|----------|----------|----------|----------|----------|-------------|-----------|----------|----------|----------|--------------|
|               | 1          | 2        | 3        | 4        | 5        | 6        | 7        | 8        | 9           | 0         | totals   |
| 1             | 45         | 0        | 0        | 0        | 0        | 1        | 0        | 3        | 0           | 49        | 91.8     |
| 2             | 0          | 50       | 1        | 0        | 0        | 2        | 0        | 2        | 0           | 1         | 55       |
| 3             | 1          | 45       | 1        | 0        | 1        | 1        | 0        | 2        | 0           | 53        | 84.9     |
| 4             | 1          | 0        | 0        | 47       | 0        | 2        | 4        | 0        | 0           | 0         | 54       |
| 5             | 0          | 0        | 0        | 0        | 48       | 0        | 0        | 0        | 0           | 48        | 100      |
| 6             | 1          | 0        | 0        | 0        | 0        | 44       | 0        | 2        | 0           | 1         | 45       |
| 7             | 1          | 1        | 2        | 1        | 0        | 0        | 39       | 1        | 0           | 0         | 45       |
| 8             | 0          | 1        | 0        | 0        | 0        | 2        | 0        | 52       | 0           | 0         | 55       |
| 9             | 0          | 0        | 0        | 0        | 1        | 0        | 0        | 41       | 2           | 45        | 91.1     |
| 0             | 1          | 0        | 0        | 0        | 0        | 0        | 0        | 45       | 0           | 45        | 93.8     |
| Column totals | 50         | 54       | 50       | 51       | 49       | 45       | 57       | 46       | 49          | 500       |
| Miss rate (%) | 10.0       | 7.4      | 10.0     | 7.8      | 2.0      | 14.3     | 8.8      | 10.9     | 8.2          | 454       |

Table 6: Confusion matrix obtained using net input method for spoken digit recognition with noise SNR=10 dB. 500 unused noisy samples were used for this test. Overall accuracy rate was 70%, calculated by summing along diagonal to count number or correct answers and dividing by 500. Off-diagonal row entries indicate number of misses for that target. Off-diagonal column entries indicate number of detection errors. Bottom right entry is diagonal total.

| Desired digit | Recognized |          |          |          | Row totals | Hit rate (%) |
|---------------|------------|----------|----------|----------|-----------|--------------|
|               | 1          | 2        | 3        | 4        | 5        | 6        | 7        | 8        | 9           | 0         | totals   |
| 1             | 38         | 0        | 1        | 3        | 2        | 1        | 3        | 0        | 3            | 59        | 64.4     |
| 2             | 1          | 35       | 10       | 1        | 0        | 5        | 2        | 2        | 1            | 59        | 59.3     |
| 3             | 2          | 6        | 34       | 0        | 0        | 2        | 3        | 5        | 3            | 56        | 60.7     |
| 4             | 2          | 1        | 2        | 39       | 0        | 8        | 4        | 3        | 0            | 2         | 61       |
| 5             | 1          | 0        | 0        | 0        | 45       | 0        | 1        | 0        | 3            | 52        | 86.5     |
| 6             | 0          | 3        | 2        | 2        | 0        | 25       | 0        | 2        | 0            | 1         | 35       |
| 7             | 4          | 3        | 3        | 1        | 0        | 0        | 25       | 0        | 2            | 41        | 68.3     |
| 8             | 0          | 2        | 2        | 1        | 0        | 5        | 1        | 35       | 0            | 0         | 46       |
| 9             | 4          | 0        | 0        | 0        | 2        | 0        | 1        | 0        | 32           | 1         | 40       |
| 0             | 2          | 1        | 0        | 3        | 1        | 2        | 1        | 1        | 0            | 40        |
| Column totals | 54         | 51       | 54       | 50       | 50       | 48       | 44       | 48       | 51           | 50      |
| Miss rate (%) | 29.6       | 31.4     | 37.0     | 22.0     | 10.0     | 47.9     | 36.4     | 27.1     | 37.3         | 20.0     | 351      |
both male and female speech signals. A signature-based classifier obtained 91% and 77% overall accuracy rates in categorizing the clean and noisy spoken digits, respectively.

Biological networks at least in part use temporal spike codes, as exemplified by the spike signatures we have generated in the present study. As mentioned in the introduction, the outputs of our network can be used as inputs for further processing such as to identify common digit sequences (e.g., "911") or more general modules for word-phrase processing.

Therefore, the proposed network is a useful architecture of spiking neurons to extract feature sets to be used for classification problem. In the future investigations, this network can be assigned as the first layer (component) of a spiking deep neural network when its feature maps are used as an input set of spike trains for the next layers of spiking neurons. Lastly, the limitation of the proposed method concerns the trade off with its complexity and performance. The small feature vector, which is used in this research, eliminates the details and would not be a convenient detector for complicated problems.

Our work also gives a useful comparison to reservoir-based methods which have achieve essentially perfect performance on this problem. In place of the reservoir, our model used a single layer spiking network equipped with STDP learning. It experimentally shows the benefit of a reservoir. At the same time, it clearly shows what can be achieved within a single layer model.

References

[1] N. Caporale and Y. Dan. Spike timing-dependent plasticity: a Hebbian learning rule. *Annual Reviews of Neuroscience*, 31:25–46, 2008.
[2] N. Kasabov, K. Dhole, N. Nuntalid, and G. Indiveri. Dynamic evolving spiking neural networks for on-line spatio and spectro-temporal pattern recognition. *Neural Networks*, 41:188–201, 2013.
[3] A. Kasinski and F. Ponulak. Comparison of supervised learning methods for spike time coding in spiking neural networks. *International Journal of Applied Mathematics and Computer Science*, 16:101–113, 2006.
[4] J. Storck, F. Jakel, and G. Deco. Temporal clustering with spiking neurons and dynamic synapses: towards technological applications. *Neural Networks*, 14(3):275–285, 2001.
[5] S. G. Wysoski, L. Benuskova, and N. Kasabov. Evolving spiking neural networks for audiovisual information processing. *Neural Networks*, 23(7):819–835, 2010.
[6] C. Panchev and S. Wermter. Spike-timing-dependent synaptic plasticity: from single spikes to spike trains. *Neurocomputing*, 58:265–371, 2004.
[7] J. Wang, A. Belatreche, L. Maquire, and T. M. McGinnity. An online supervised learning method for spiking neural networks with adaptive structure. *Neurocomputing*, 144:526–536, 2014.
[8] T. Masquelier and S. J. Thorpe. Unsupervised learning of visual features through spike timing dependent plasticity. *PLoS Computational Biology*, 3(2):247–257, 2007.
[9] T. Masquelier and S. J. Thorpe. Learning to recognize objects using waves of spikes and spike-timing-dependent plasticity. In *The 2010 International IEEE Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2010.
[10] S. G. Wysoski, L. Benuskova, and N. Kasabov. Fast and adaptive network of spiking neurons for multi-view visual pattern recognition. *Neurocomputing*, 71(13):2563–2575, 2008.
[11] J. J. Wade, L. J. McDaid, J. Santos, and H. M. Sayers. SWAT: a spiking neural network training algorithm for classification problems. *IEEE Transactions on Neural Networks*, 21(11):1817–1830, 2010.
[12] J. Pflüster, T. Toyoizumi, D. Barber, and W. Gerstner. Optimal spike-timing-dependent plasticity for precise action potential firing in supervised learning. *Neural Computation*, 18(6):1318–1348, 2006.
[13] L. R. Rabiner and J. Biing-Hwang. *Fundamentals of Speech Recognition*. Prentice Hall, 1993.
[14] A. Dibazar, D. Song, W. Yamada, and T. W. Berger. Speech recognition based on fundamental principles of the brain. In IEEE 2004 International Joint Conference on Neural Networks, pages 3071–3075, 2004.

[15] C. Näger, J. Storck, and G. Deco. Speech recognition with spiking neurons and dynamic synapses: a model motivated by the human auditory pathway. Neurocomputing, 44:937–942, 2002.

[16] H. H. Narmavar, J. S. Liaw, and T. W. Berger. A new dynamic synapse neural network for speech recognition. In IEEE 2001 International Joint Conference on Neural Networks, pages 2985–2990, 2001.

[17] D. Verstraeten, B. Schrauwen, D. Stroobandt, and J. V. Campenhout. Isolated word recognition with the Liquid State Machine: a case study. Information Processing Letters, 95(6):521–528, 2005.

[18] A. A. Dibazar, H. H. Namarvar, and T. W. Berger. A new approach for isolated word recognition using dynamic synapse neural networks. In IEEE 2003 International Joint Conference on Neural Networks, pages 3146–3150, 2003.

[19] S. Louiselle, J. Rouat, D. Pressnitzer, and S. Thorpe. Exploration of a rank order coding scheme with spiking neural networks for speech recognition. In IEEE 2005 International Joint Conference on Neural Networks, pages 2076–2080, 2005.

[20] D. Verstraeten, B. Schrauwen, and D. Stroobandt. Reservoir-based techniques for speech recognition. In Proc 2006 Intl Joint Conf on Neural Networks, pages 1050–1052, Vancouver, July 2006.

[21] B. Schrauwen, J. Defour, D. Verstraeten, and J. V. Campenhout. The introduction of time-scales in reservoir computing, applied to isolated digits. In Artificial Neural Networks – ICANN 2007, pages 471–479. Springer, 2007.

[22] M. D. Skowronski and J. G. Harris. Automatic speech recognition using a predictive echo state network classifier. Neural Networks, 20(3):414–423, 2007.

[23] A. Ghani, M. McGinnity, L. P. Maquire, and J. Harkin. Neuro-inspired speech recognition with recurrent spiking neurons. In V. Kurkova-Pohlova and J. Koutnik, editors, Artificial Neural Networks – ICANN 2008, pages 513–522. Springer, 2008.

[24] M. Lukoševičius and H. Jaeger. Reservoir computing approaches to recurrent neural network training. Computer Science Review, 3(3):127–149, 2009.

[25] T. Masquelier, R. Guyonneau, and S. J. Thorpe. Competitive stdp-based spike pattern learning. Neural Computation, 21:1259–1276, 2009.

[26] S. Klampfl and W. Maass. Emergence of dynamic memory traces in cortical microcircuit modes through STDP. Journal of Neuroscience, 33(28):11515–11529, 2013.

[27] A. V. Oppenheim, R. W. Schafer, and J. R. Buck. Discrete-time signal processing. Prentice-hall, 1989.

[28] A. Tavanaei, M. T. Manzuri, and H. Sameti. Mell-scaled discrete wavelet transform and dynamic features for persian phoneme recognition. In IEEE Symposium on Artificial Intelligence and Signal Processing, pages 138–140, Tehran, Iran, June 2011.

[29] E. M. Izhikevich. Simple model of spiking neurons. IEEE Transactions on Neural Networks, 14(6):1569–1572, 2003.

[30] D. Pearce and H. G. Hirsch. The Aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions. In Automatic Speech Recognition: Challenges for the New Millenium, pages 181–188, Paris, France, 2000.