Real-time Anomaly Detection for Streaming Data using Burst Code on a Neurosynaptic Processor

Qiuwen Chen, Qinru Qiu
Department of Electrical Engineering and Computer Science, Syracuse University, NY 13244, USA
Email: {qchen14, qiqiu}@syr.edu

Abstract*—Real-time anomaly detection for streaming data is a desirable feature for mobile devices or unmanned systems. The key challenge is how to deliver required performance under the stringent power constraint. To address the paradox between performance and power consumption, brain-inspired hardware, such as the IBM Neurosynaptic System, has been developed to enable low power implementation of large-scale neural models. Meanwhile, inspired by the operation and the massive parallel structure of human brain, carefully structured inference model has been demonstrated to give superior detection quality than many traditional models while facilitates neuromorphic implementation. Implementing inference based anomaly detection on the neurosynaptic processor is not straightforward due to hardware limitations. This work presents a design flow and component library that flexibly maps learned detection network to the TrueNorth architecture. Instead of traditional rate code, burst code is adopted in the design, which represents numerical value using the phase of a burst of spike trains. This does not only reduce the hardware complexity, but also increases the results accuracy. A Corelet library, NeoInfer-TN, is developed for basic operations in burst code and two-phase pipelines are constructed based on the library components. The design can be configured for different tradeoffs between detection accuracy and throughput/energy. We evaluate the system using intrusion detection data streams. The results show higher detection rate than some conventional approaches and real-time performance, with only 50mW power consumption. Overall, it achieves 10^8 operations per watt-second.

I. Introduction

With the blooming of machine learning and neural networks, intelligent systems have been developed for various applications such as image recognition [16], multi-media retrieval [5] and intrusion detection [17]. Many of them process streaming data in real-time and imposes high demand in accuracy and computation throughput. Real-time anomaly detection is one of these applications that continuously monitors and processes incoming data streams for patterns that do not conform to normality. It is an extremely desirable feature for improving the autonomy of today's unmanned systems or mobile devices. However, to deliver the required performance under limited power constraint is a major design challenge.

The brain has unprecedented performance and energy efficiency in cognitive tasks [10]. It is believed that perception is a procedure of probabilistic inference, and the efficiency of the biological neural system comes from its massive parallel architecture, spiking based communication and closely coupled computing and storage. Inspired by the biological structure, the IBM Neurosynaptic System with the TrueNorth architecture [12] provides a low-power platform for large-scale prototyping of Spiking Neural Network (SNN) based intelligent systems. Meanwhile, brain-inspired anomaly detection has been proposed [7] and shown to give superior detection quality than many traditional approaches. It performs inference based detection and features massive parallel structure that facilitates neuromorphic implementation. Despite the detection quality and fast processing, the system consumes high power, and migrating the model to an SNN will provide huge optimization for power efficiency. However, implementing the inference network on the TrueNorth processor is not straightforward.

A TrueNorth chip contains 4096 neurosynaptic cores, each of which has 256 axon inputs and 256 neurons. The synaptic connectivity is realized by a 256x256 crossbar. The connected core networks are encapsulated into Corelet [1] for abstraction and modular designs. While the processor provides potential to address the performance and power constraints for real-time embedded applications, challenges exist when mapping a signal processing flow onto this platform due to its hardware constraints. Firstly, each neuron (column) can only support 4 different input weights, while the weight of a connection is decided by the axon type (row). This limits all neurons who share an axon input to use the same weight rank at that row. However, most models' learned parameters are real numbers. Secondly, neurons communicate with each other using spikes, how to encode numerical value into spike trains is application specific. Thirdly, the crossbar's size of a core constrains the fan-in and fan-out of a neuron to 256. This hinders the direct mapping of big networks. Finally, some common arithmetic operations, such as division and maximum, are not as readily supported in TrueNorth as in traditional architectures.

In this work, we focus on implementing a trained inference network on TrueNorth for real-time anomaly detection [7]. Instead of rate code, which has been widely used in many other TrueNorth applications [8], burst code is used because it gives higher computation accuracy and allows very simple implementation of certain arithmetic operations, such as maximum. A Corelet library is developed, which consists of neural circuits for operations in the anomaly detection model. Our system first extract the topology and parameters of the network from the learned knowledge base. Then, it flattens and maps the network to the Corelet library components for TrueNorth configuration. With a controllable clock driver input, the network is activated by streaming data in an event-driven way. Anomaly scores are calculated in real-time by probabilistic inferences of spatial-temporal features. To our best knowledge, this is the first work that applies neurosynaptic processor to the real-time anomaly detection.

The contributions of this paper are the followings:

- We build a generic parser that transforms and maps

This work is partially supported by the National Science Foundation under Grants CCF-1337300.

inference-based anomaly detection network [6], [7] to a spiking neural network.

- A novel spike burst coding scheme is proposed for efficient representation, high accuracy and more convenient implementations. A Corelet library, *NeoInfer-TN*, is developed, which contains the neural circuit implementation of the network components for this coding.
- The adoption of bust code enables a two-phase pipelined processing for higher throughput. The throughput only depends on the spike encoding window configured to achieve a required data precision.
- A tunable accuracy factor is provided to enable tradeoff between detection accuracy and throughput.
- The network is evaluated with real-time intrusion detection data stream, and the accuracy, throughput and power performance are reported.

II. Detection Algorithm

Inference-based anomaly detection considers the anomaly as an unexpected observation in a given environmental context. It calculates the likelihood of each observation and at the same time infer the most likely one from the context. If the likelihood of the real observation is much lower than that of the expected, then an alarm is raised. The detailed detection algorithm and analysis were elaborated in Chen et. al [6], [7], so this section only sketches the computation procedure.

Cogent confabulation [9] is adopted the computing model for probabilistic inference. Cogent confabulation describes an application using a set of features (e.g. color and shape). The observed attributes of a given feature (e.g. red color, round shape) are represented as neurons, and their pairwise conditional probabilities are represented as synapses. To better organize the knowledge, neurons that represent the same features are grouped into *lexicons*, and the synapses between the neurons of two lexicons are realized as a probability matrix. The i, j^{th} entry of such matrix gives the conditional probability $p(s_i|t_j)$ between neuron s_i in the source lexicon and t_j in the target lexicon.

Whenever an attribute is observed, the corresponding neuron is activated, and its excitation is passed to the other neurons through the synapses. The excitation of a neuron t in lexicon l is calculated by summing up all incoming activations:

$$y(t) = \sum_{k \in F_l} \{ \sum_{s \in S_k} [I(s) \ln \frac{p(s|t)}{p_0}] \}, t \in S_l \tag{1}$$

Here, F_l denotes the set of lexicons that have connections to l, and S_k is the set of neurons in lexicon k. $I(s)$ takes 0 or 1 given the activation of neuron s. p_0 is a constant selected to ensure positive weights. The excitation is essentially the log likelihood of t given the rest of the observations.

A set of lexicons are selected and referred to as *key lexicons*. They are the testing units of abnormality, representing features of the possible observations. The other lexicons that are not tested are named *support lexicons* and they represent the environmental context. Synapses are established from the support lexicons to the key lexicons. The excitations of all neurons in a key lexicon are calculated using function (1). The neuron with the highest excitation is considered the reference neuron, denoted $t_{\max}$. The *anomaly score* of a key lexicon l is computed using Equation (2).

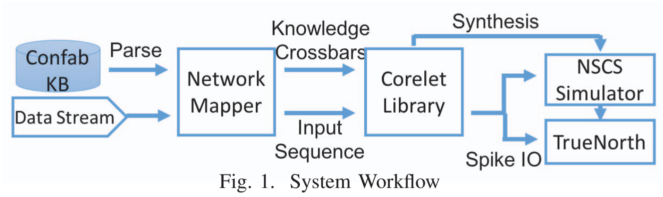


Fig. 1. System Workflow

$$\alpha_l(t) = \frac{y(t_{\max}) - y(t)}{y(t_{\max})}, t, t_{\max} \in S_l \tag{2}$$

Here, the anomaly score is the normalized difference between the observed activation at t and the prediction $t_{\max}$. It indicates how low the sample's cogency is given the context. The scores of all lexicons are merged into the network anomaly score:

$$A(t_{l=1\ldots L}) = \frac{\sum_{l=1}^{L} \alpha_l(t_l)}{L} \tag{3}$$

L is the number of key lexicons. t_l is the observation neuron of lexicon l. The output score is ranged in $[0, 1]$.

The efficiency of confabulation-based anomaly detection comes from a carefully structured inference network. We use the self-structuring method [7] to build the feature hierarchy in the lexicon design space, which generates a succinct network.

III. System Design

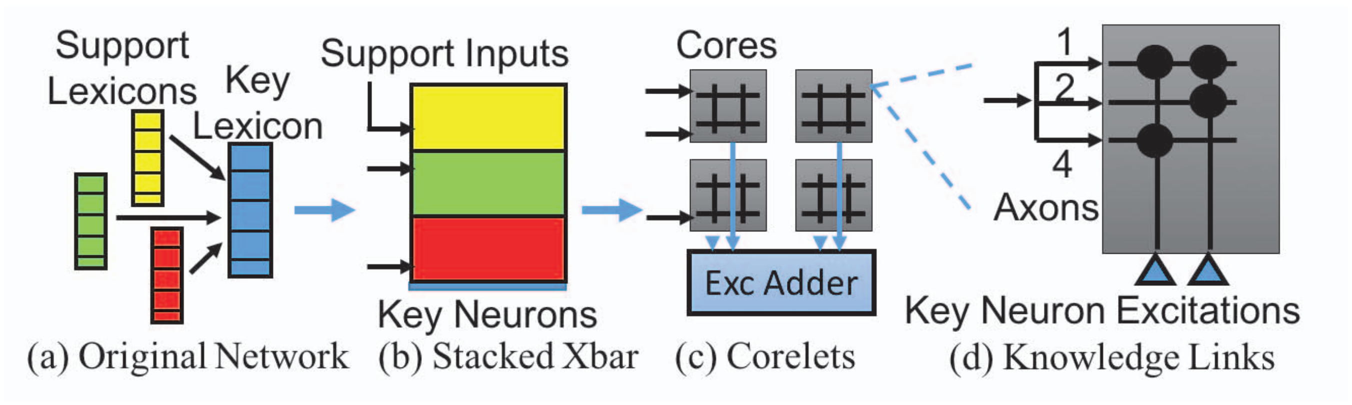
The aforementioned detection flow has four layers, (a) the support lexicon layer that collects input from the environment, (b) key lexicon layer that calculates neuron excitations using Equation (1), (c) anomaly score generator, which performs Equation (2) for each key lexicon, (d) and anomaly score accumulator, which merges all key lexicon anomaly scores using Equation (3). To convert the network into Corelets on TrueNorth, the overall workflow is shown in Fig. 1.

The network mapper reads in a trained confabulation knowledge base (KB) and maps the connections between support and key lexicons to a set of crossbar matrices considering the hardware constraints. It also maps the neuron observations to the input of the processor and synthesizes the input spikes from the given data stream. In the second step, the synthesizer maps each crossbar matrix into Corelets using our NeoInfer-TN library in the Corelet Programming Environment (CPE). It also maps the score generation and score accumulation layers into library components. At last, the network and its inputs are tested on the NSCS simulator [15] and the TrueNorth chip.

A. Network Mapping

The connection between neurons in the support and key lexicons forms a bipartite graph and can naturally be implemented as crossbar arrays, where each row corresponds to a neuron in the support lexicon and each column to a neuron in the key lexicon. However, a core can only implement crossbar up to 256x256, while the number of neurons in the support and key lexicons can reaches 3000. Matrix partition must be considered to implement one connection using multiple cores.

How to accurately represent synaptic weight in the crossbar is another challenge. Due to the hardware limitation, each column can only support 4 different weights, and all weights in


Support Lexicons
Key Lexicon
Support Inputs
Key Neurons
Cores
Exc Adder
1
2
4
Axons
Key Neuron Excitations
(a) Original Network
(b) Stacked Xbar
(c) Corelets
(d) Knowledge Links