Self-structured Confabulation Network for Fast Anomaly Detection and Reasoning

Qiuwen Chen*, Qing Wu†, Morgan Bishop†, Richard Linderman† and Qinru Qiu*
* Department of Electrical Engineering and Computer Science, Syracuse University, NY 13244, USA
Email: {qchen14, qiqiu}@syr.edu
† Air Force Research Laboratory, Information Directorate, RITC, 525 Brooks Road, Rome, NY, 13441, USA
Email: {Qing.Wu, Morgan.Bishop, Richard.Linderman}@rl.af.mil

***Abstract*—Inference models such as the confabulation network are particularly useful in anomaly detection applications because they allow introspection to the decision process. However, building such network model always requires expert knowledge. In this paper, we present a self-structuring technique that learns the structure of a confabulation network from unlabeled data. Without any assumption of the distribution of data, we leverage the mutual information between features to learn a succinct network configuration, and enable fast incremental learning to refine the knowledge bases from continuous data streams. Compared to several existing anomaly detection methods, the proposed approach provides higher detection performance and excellent reasoning capability. We also exploit the massive parallelism that is inherent to the inference model and accelerate the detection process using GPUs. Experimental results show significant speedups and the potential to be applied to real-time applications with high-volume data streams.**

I. Introduction

Anomalous data stream detection is inherently hard. Firstly, labeled data are expensive to obtain, and some abnormal classes are not foreseeable by the time of modeling. Secondly, the input streams are continuous and infinite, which requires the model to learn new data by one pass and recall in a real-time manner. Thirdly, users require more than just output labels, so the algorithm should also provide insights into the decision-making process. Such restrictions rule out many traditional methods such as multi-class classifiers, off-line analysis and obscure models.

Studies [8], [9] suggest that manually configured confabulation networks [14] can effectively detect outliers and provide reasoning in some well understood problems. However, building such networks might require expert knowledge. Thus the neuron nodes (i.e. lexicons) and the synapses (i.e. knowledge links) between them must be re-configured when applied to new applications. This limitation makes the confabulation-based approaches inflexible in handling different datasets. To address the above problems, we present AnRAD (Autonomous Anomaly Reasoning and Detection), a transparent framework that provides real-time online detection using a self-structured confabulation network.

To motivate the discussion about transparent network and self-structuring, consider an example of detecting voice recordings spoken in foreign languages. An artificial neural network can be used to identify a non-English clip. It encodes vocal features into nodes through weighted links and the weight adjustment is achieved by repeated practice of native tongues. Although such detection is fast, it does not reveal why the clip is not in English since the nodes lose the original meanings. In contrast, a confabulation network preserves the meaning of the features, and therefore, it might reveal some inconsistencies of tone combinations in the English context. Although the recall process might be slightly slower than that of a neural network, it provides valuable information about why a tested subject is labeled as an anomaly. Unlike neural networks that employs fixed numbers of nodes and links, which nodes to select and how to connect them together in the confabulation network is application specific, and hence usually requires domain knowledge from the experts.

In this work, we present a self-structured confabulation network model, whose configuration is learned from an initial set of data. The proposed self-structuring technique concretely learns from the data a succinct set of nodes that represent original features or combinations of features. These nodes are named lexicons because they record the symbolic representations of the possible inputs. The links between nodes are also learned from the initial data. Given the learned network configuration, further incoming data streams are used to incrementally refine the knowledge links to calculate the conditional probability between the lexicon symbols. The learned knowledge bases are accessed in the recall phase to test the inconsistency of each node (Section III). A GPU-based parallel implementation is adopted to achieve computation acceleration. The proposed framework is generalized to a wide range of applications. In this work, it is applied to road-traffic monitoring, network intruder detection and program control flow as case studies.

The key contribution of this work is an automatic procedure that learns the structure of a confabulation network from the incoming data (Section IV). The constructed model consists of well-defined nodes that capture both spatial and temporal relations among the features of the dataset. Also, a method of incremental model refinement is proposed, which effectively constructs the knowledge base on-the-fly using the incoming data streams (Section IV-E). The detection performance is then compared with those of classical methods in Section V. Last but not least, the recall process is implemented on a GPU to provide real-time detection capabilities (Section VI).

II. Related Work

Extensive studies on anomaly detection [7] have been carried out. In principle, they can be divided into four categories: classification-based, nearest-neighbor-based, cluster-based and statistical methods. The classification-based methods learn a classifier from labeled training data and classify a test subject into one of the classes. Examples include SVM, rule-based and neural networks. However, the classification-based detector requires labeled data as the training set, which limits their applicability in unsupervised scenarios. Self-organized map (SOM) [6] and replicator neural networks [13] were studied to remedy the dependency on training labels. However, they still cannot avoid the obscurity of neural network models. The nearest-neighbor-based detector assumes normal data occurs in dense neighborhoods while anomalies are far from their neighbors. The relative density has been defined in local distance (LOF) [4] or rank [15]. To remove the off-line limitation and reduce the complexity, an incremental variation of LOF was developed [20]. But the recall complexity of the method scales with training sample size, which makes it unable to handle large data sets or streams. The cluster-based techniques are unsupervised, and assume normal instances belong to a cluster in the data while anomalous ones do not [12]. However, it also suffers from high complexity and does not support on-line training. The statistical models fit the data with parametric distributions and considerable anomalies occur in the low probability regions [1]. But they cannot handle streams with varying distributions. Finally, all of the above methods consider testing the subjects as single data points. When applied to detecting abnormal data streams, they all require moving-window-based or segmentation-based preprocessing on the data stream.

Cogent confabulation [14] has been applied to sentence completion [2], [21] and document image recognition in previous studies. The theory exploits the concept of distributed and symbolic information representations of the brain [22], and is applied to inference-based cognition. One previous work extends the theory to abnormal road traffic detection [8]. But the manually configured model lacks the flexibility to be extended to other data sets. To our best knowledge, there is no prior study in self-structured confabulation network dedicated for anomalous stream detections. Also, there is a challenge to provide real-time detection with a transparent network. Although the combination of machine learning and high performance computing (HPC) has received wide attentions [3], [23], this has not been explored for confabulation-based anomaly detection applications.

III. Confabulation-based Anomaly Detection

AnRAD is a confabulation-based anomaly detection framework. Given an input, test is carried on each key node. Each test calculates an anomaly score reflecting how much the observed input deviates from general experiences. The anomaly score is determined by the excitation levels of lexicon symbols computed based on the cogent confabulation principle, hence symbolic level parallelism can be achieved. This section explains how the single test nodes work together to obtain a network anomaly score.

Cogent confabulation is a connection-based cognitive model that captures correlations between features at the symbolic level. In this model, the observed attributes (e.g. black color, 30mph speed) are referred to as symbols, and their pairwise conditional probabilities are referred to as knowledge links. Symbols are analogous to neurons in biological nervous system, and knowledge links are analogous to synapse plasticity between neurons. For better organizing the knowledge, neurons that represent the same features (e.g. colors, speeds) are grouped into lexicons, and links between lexicons are realized as probability matrices. Lexicons and the knowledge links between lexicons form a knowledge graph; therefore, we also refer to lexicons as nodes. During learning, these knowledge links are established and strengthened as symbols being co-activated. Given new observations, familiar information with high relevancy will be recalled from the knowledge base. This relevancy measure, called excitation level, is calculated by the following function (1).

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

In this function, t is one of the symbols of node S_l to recall; F_l denotes the set of nodes that have connections to l, and S_k is the symbol set of lexicon k; $I(s)$ is the firing strength of source symbol s; p_0 is the minimum probability that considered informative; B is a constant band gap to favor symbols receiving more activation from distinct lexicons. The excitation level is essentially the log likelihood of symbol s given the rest of the observations.

Originally, the confabulation model is used to infer the most likely symbols. This work, however, applies the theory in a reverse way. To start with the model, application's dimensional features are defined as lexicons, within which those having incoming connections are called *"key lexicons"* and serve as the basic testing units. The other lexicons without incoming connections will not be tested, and are named *"supporting lexicons"*. The excitation levels of all possible symbols in a key lexicon is calculated according to function (1). The symbol with the highest excitation (i.e. the highest likelihood) is considered the reference symbol. An anomaly score is calculated for the observed input symbol using the following function (2).

$$as_l(v) = \frac{ref_{t \in S_l}(t) - el(v)}{ref_{t \in S_l}(t)}, v \in S_l \qquad (2)$$

The score is the normalized excitation difference between observation $el(v)$ and reference symbol $ref_{t \in S_l}$. It reflects how low the observed symbol's cogency is compare to its context. In this way, all key lexicons have their individual anomaly score calculated and accumulated to a network anomaly score. If the input is truly abnormal, the score is expected to be high, as many nodes would show deviations. The network anomaly score (nas) is computed by averaging the prior-modified node scores as_l^*. In equation (3), L is the number of key lexicons, and the output score is ranged in $[0, 1]$.

$$nas(v_{l=1...L}) = \frac{\sum_{l=1}^{L} as_l^*(v_l)}{L} \qquad (3)$$

$$as_l^*(v \in S_l) = \frac{(ref_{t \in S_l}(t) + lp(t)) - (el(v) + lp(v))}{ref_{t \in S_l}(t) + lp(t)} \qquad (4)$$

where $lp(v) = ln(pr(v)/p_0)$ is the logarithic prior probability of symbol v. The workflow reveals a promising approach to fulfill the requirements for anomalous stream detection, but its performance essentially depends on the selection of key lexicons and their incoming links. To construct the confabulation network requires application specific knowledge. Expert

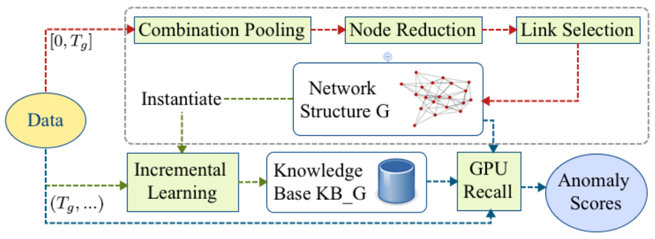


Fig. 1. AnRAD workflow

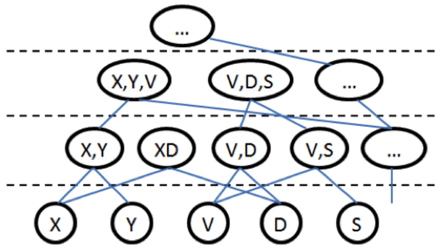


Fig. 2. Hierarchical Structure Example

knowledge may not always be available, and it does not always ensure optimal network structure. Self-structuring then plays an important role to improve the framework's generality and applicability.

IV. Model-learning Algorithms

In the AnRAD framework, inputs can be represented as data stream $\{x^1, x^2, ..., x^t, ...\}$, generated from some distribution D. Here x^t represents a record tuple at time frame t, and consists of Q features denoted by $x^t(q)$. As shown in Fig. 1, in the structuring stage, a span of the data at frame $[0, T_g]$ is sampled and used to construct the hierarchical confabulation network G that best describes the application. The strucuring procedure has three components: combination pooling, node reduction and link selection. After the network is constructed, new streams at $(T_g, T_0]$ are used to train the initial knowledge bases $KB_G^{T_g:T_0}$. The knowledge bases are applied to streams at $(T_0, ...)$ to generate network anomaly scores for each frame. At the same time, the new incoming data continuously refine the knowledge bases $KB_G^{T_g:t}$. Typically, a moving window with size W, $\{x^{t-W}, ..., x^{t-1}, x^t\}$ is applied to the input stream at frame t to select the data for processing. This section addresses the generation of network G and refinement of knowledge base $KB_G^{T_g:t}$.

A. Key Node Hierarchy

The confabulation model can only capture the first order relation between features. A higher order relation has to be considered by adding new lexicons corresponding to feature combinations. The final structure of confabulation network consists of hierarchical lexicons where higher-level nodes are formed as the compositions of lower-level nodes as shown in Fig. 2. Lexicons at the bottom layer represent single primary features. These primary features provide a basic description of the input data. The higher-level lexicons assemble multiple primary features; they represent more abstract meanings and combinational patterns. The layered structure provides direct mapping from the feature space to nodes in the knowledge graph, but its complexity may increase exponentially if implemented naively. Since the confabulation network works at the symbolic level, continuous features are discretized before mapping to symbols in lexicons. Given the composition of features, higher-level lexicons have coarser discretization intervals than lower level lexicons. For the study cases in this work, the equal-width bins are used for the primary features. The higher-level lexicons have each of their component features discretized with bins two times wider than in the previous lower orders. More sophisticated method may be developed to further improve the performances.

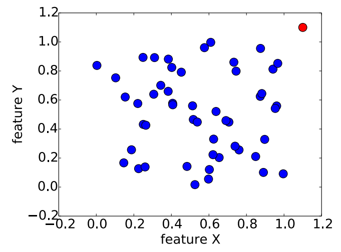
Note that each input data point is a segment of the data stream within a time span. The primary features may also have a timestamp. The composition of features does not only happen spatially but also temporally. For example, there may be a feature composition $\langle x_n^t(q), x_n^t(q') \rangle, q, q' \in Q$, or $\langle x_n^t(q), x_n^{t-\Delta t}(q) \rangle, \Delta t < W$. Hence, temporal relations among them are also learned and checked.

A question raised here is which feature combinations should be included. If the model simply considers all possibilities, it would quickly scale to an intractable lexicon size as Q and W increase. This is both unnecessary and computationally wasteful. Another option is to rely on traditional feature reduction techniques. Although these techniques have long been studied, they either require supervised learning [17], or destroy the direct relation (one-to-one mapping) between feature space and lexicon space (principal component analysis [16]). None of the previous techniques have been applied to feature combinations. To solve the problem, we propose a pool-reduction procedure that is applied in both spatial and temporal domains to construct the key lexicon's hierarchy. Furthermore, a link selection algorithm is also presented to find the connections between the nodes and their supports.

B. Feature Combination Pooling

As mentioned previously, we complement the primary features with a set of composite features to capture higher order associations. We refer this step as feature pooling. The pooling stage generates a set of lexicon candidates, which will be reduced as discussed in the next section.

Take a simple two-feature combination for instance, the first question to ask is whether such combination provides more information for anomaly detection than the individual feature components. Consider the example scatter plot in Fig. 3, where the X and Y axes represent the dimensions of the two primary features. If the two features are distributed independently in their feature space as those blue dots in Fig. 3a, a potential outlier (the red dot) in this subspace can be detected by considering only one of the components. Therefore combinations of non-correlated features do not offer additional information. However, if the two features are sufficiently relevant as shown in Fig. 3b, the red dot, which is originally indistinguishable from any single axis, will be detected by their combination. Based on this observation, the pooling procedure is design to keep the combination of highly correlated features.


(a) Uncorrelated Components

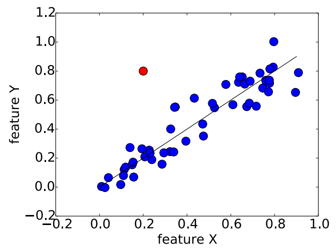

(b) Correlated Components

Fig. 3. Relevant Feature Example

To extend this concept to more general cases, we define the feature distance $d(q_i, q_j) = [1 - MI(q_i, q_j)] \in [0, 1]$, where $MI(.)$ calculates the normalized mutual information between the two vectors. The smaller the distance is, the more relevant the two features q_i and q_j are. For combination Q_l consisting of two or more features, a simple relevancy test is performed to determine whether it is included in the lexicon candidate set:

$$RT(Q_l) = \prod_{q_i, q_j \in Q_l} I[d(q_i, q_j) < d_{prox}] \quad (5)$$

This test requires all the component pairs in Q_l to be sufficiently close to each other. And d_{prox} is a constant proximity distance that defines the largest distance that is considered relevant. Algorithm 1 is used to pool the features for lexicon generation. The algorithm first adds all the single features into the candidate set. Then in the second for-loop, each subset of Q whose cardinality is less than *max_order* is inspected. If the subset passes the relevancy test, a new lexicon candidate will be added for it. Not all candidates will be key lexicons whose anomaly score will be calculated. A reduction stage is used to select the key lexicons from the candidate set.

Algorithm 1 Feature Combination Pooling

```
1: procedure pool($Q$, max_order):    # $Q$: the complete feature set; max_order: the maximum combination order
2: $CS \leftarrow empty\ set$
3: for each feature $q \in Q$:
4:     add $\{q\}$ to $CS$
5: for each $Q_l \subset Q$ and $|Q_l| <$ max_order:
6:     if all $Q_{l'} \subset Q_l$ was accepted and $RT(Q_l)$ passed:
7:         add $Q_l$ to $CS$
8: return $CS$    # feature combination candidate set
```

C. k-NN Node Reduction

Although the pooling process excludes most of the irrelevant combinations, the number of possible candidates may still be large if Q has many features. Therefore, a reduction procedure is used to further compress the candidate set to generate key lexicons. The redundancy among candidates selected in the pooling stage should be removed during the reduction. Because labels are not available in the training set, a similarity-based method [19] is modified to preserve the most representative combinations.

The general idea of the reduction procedure is to cluster the candidate feature combinations by their similarity, and then select one representative from each of the clusters. Again, normalized mutual information is employed to measure the distance $d(Q_{l1}, Q_{l2})$ between the combinations. The clustering process is accomplished by k-NN (k nearest neighbor) principle. While the most compact candidate is selected from a cluster, its neighbors will be discarded. This operation repeats until the remaining candidates cannot form any cluster. The reduction procedure is described in Algorithm 2. The algorithm first initializes the set *KEY* with all candidates. Then it calculates the distances from each combination to its nearest neighbors. The center of the compact cluster has its k-distance selected as the upper limit of cluster radius. Then in the following while-loop, the combination with minimum k-distance is selected and has its K neighbors removed from the *KEY* set. Then the K value is reduced until the next cluster would have a smaller radius than the radius limit. The neighbor-removing process repeats until K reaches 1. The remaining candidates in *KEY* set are the final nodes selected.

Algorithm 2 Node Selection

```
1: procedure knn($CS$, $K$):    # $CS$: pooled candidate set; $K$: the initial K
2: KEY $\leftarrow CS$
3: for each combination $Q_l \in CS$:
4:     for $Q_k \in K$ nearest neighbor of $Q_l$:
5:         $Q_l$.dist[k] $\leftarrow d(Q_l, Q_k)$
6: find $Q_0$ who has the smallest k-distance
7: max_err $\leftarrow Q_0$.dist[$K$-1]
8: while $K > 1$:
9:     find $Q_r$ who has the smallest k-distance
10:    remove $Q_r$ 's $K$ nearest neighbor from KEY
11:    radius $\leftarrow$ min($Q_x$.dist[$K-1$]) for $Q_x \in$ KEY
12:    while radius > max_err:
13:        $K \leftarrow K - 1$
14:        radius $\leftarrow$ min($Q_x$.dist[$K-1$]) for $Q_x \in$ KEY
15: return KEY    # key node set
```

Although we use the features in Q as an example to explain the pooling-reduction procedure, the concept in Section IV-B and IV-C can be applied to temporal domain as well. When the data inputs are not just single points in the feature space but multi-variant time series, the definition of anomalies may extends to historical patterns. To capture such potential outliers, the key lexicons must include not only different features, but also feature projections in different frames. This can be accomplished by performing feature-wise selection followed by temporal selection. If multiple frames are considered after the key lexicons being built, each lexicon along with its historical readings form a new temporal feature set $Q_l^W = \{Q_l^0, Q_l^{-1}, ..., Q_l^{-W}\}$. The same pooling-reduction algorithms can then be applied directly on these feature sets to generate informative and succinct key lexicons. A key lexicon is represented as a two-dimensional pattern $R_l \sim [(q_{l_1}, q_{l_2}, ..., q_{l_i}, ...)^{-t_1}, (..., q_{l_i}, ...)^{-t_2}, ...(..., q_{l_i}, ...)^{-t_j}, ...]$. Furthermore, the number of correlated frames W is usually much smaller than the feature number in Q, so the reduction process may sometimes be omitted in temporal selection.

D. Link Selection

Now that the key lexicons are identified, the next step is to find the supporting lexicons that can be used to infer the key lexicon symbols. To do so we follow a max-similarity, min-redundancy principle. For instance, to infer the shape of an object, touching is preferable compared to color (max-similarity). But if touching is already selected, weighting might not be necessary as they share some information (min-redundancy). Generally, we want to maximize the correlation between key lexicons and their supporting lexicons, and meanwhile, minimize the correlation among the supporting lexicons that are connected to the same key lexicon. The supporting lexicons are chosen from the primary features since the key lexicons have already handled the combinational patterns.

Algorithm 3 Link Selection

```
1:  procedure select_links($R$, $F$):        # $R$: target node; $F$: set of single features
2:  SUPP ← empty set
3:  ranks ← $F$.sort(key=$d(R, q \in F)$)
4:  for feature $q \in$ ranks:
5:      if $q \in R$ or $d(R, q) > (1 - d_{prox})$:
6:          continue          # low similarity
7:      if any $p \in$ SUPP has $d(p, q) < d_{prox}$:
8:          continue          # high redundancy
9:      add $q$ to SUPP
10: return SUPP          # support nodes for R
```

Heuristic Algorithm 3 finds a group of features at certain time offset, $\{q^{-t}, q \in Q, t < W\}$ which infer the observation at key lexicon R_l. The algorithm first sorts the supporting features by their distances to the target key lexicon. Then it traverses the sorted features, adds a primary feature to the supporter set only if: (1) it is not one of the components of the key lexicon; (2) it is highly correlated with the key lexicon; and (3) it has low correlation with supporting features that has already been selected for the same key lexicon.

At this point, the confabulation model is properly configured, and the training streams can be processed to build the knowledge base.

E. Incremental Training

Given the configured confabulation network, AnRAD then constructs the knowledge base from the input data streams. Basically, the learning process is to find out the weight of the knowledge links, i.e. the $p(s|t)$ values in Equation 1. This can be achieved by collecting the statistics of co-occurrences of linked lexicon symbols. The probability then is calculated as $p(s|t) = cnt(s,t)/cnt(t)$, where $cnt(s,t)$ is the number of co-occurrence of the source and target symbols s and t, while $cnt(t)$ is the occurrence of target symbol t.

For inferencing tasks, above method works fine because more data samples generally give better accuracy regarding the most likely candidates. However, anomaly detection is different, in that the most unlikely candidate is not necessarily more distinguishable as the sample size increases. Actually, some studies [25] even suggest smaller datasets outperform larger ones. Constantly incrementing the co-activation counters may even degrade the detection performance. Therefore, our framework uses a mechanism named *"episodic training"*, in which the co-activation counters reset after every time period T_{ep}. Then the excitation stored in the knowledge base is updated by merging the new one into previous episodes.

$$ex^{E+1}(s,t) = \frac{ex^E(s,t) * E + ln[p(s|t)/p_0]}{E+1} \tag{6}$$

$$ex^0(s,t) = ln(\frac{p(s|t)}{p_0}) \tag{7}$$

ex^E is the stored excitation at episode E. It can be substituted into equation (1) as $el(t) = \sum_{k \in F_l}\{\sum_{s \in S_k}[I(s)ex(s,t)] + B\}$. In the case before first reset, i.e. ex^0, the result is the same as equation (1). Essentially, this updating function works as ensemble of temporal sub-samples.

V. Evaluations

To evaluate the effectiveness of the framework, we investigate three different datasets. The first dataset contains vehicle traces. The second one is composed of extracted package streams from DARPA 1998 intrusion detection dataset [18]. The last case contains system call sequences of benign and malicious programs on Linux workstations [10], [11]. Since the AnRAD framework and the baseline methods generate scores for each input frame, we used a same leaky bucket algorithm [24] with capacity 2.0 and leak rate 0.5 as the decision stage for the data streams. Whenever the score generated by a method exceeds its threshold, it fills 1 unit into the bucket. The stream is reported anomalous when the water overflows the bucket.

A. Abnormal Vehicle Behavior Detection

In this application, vehicle traces are obtained from an area road network. The preprocessor extract 10 primary features of which 5 are vehicle features (latitude, longitude, speed, direction, and vehicle type) and 5 are interactive features (the neighbor vehicle's distance, speed, relative position, direction difference and type). The self-structuring procedure picks 44 key lexicons out of 2548 possibilities given $max\_order = 5$ (feature-wise pooling) and $= 3$ (temporal pooling). The traffic records are generated at one-second sampling intervals and in four randomly picked zones. The training stage consumed 240 minutes of traces and another 10-minute trace is used as the test set. Among the test data, there are 179 vehicles without intentional modification used as negative cases, and 22 manually created anomalies of different categories are used as positive cases.

Fig. 4 shows the detection results of anomaly classes. The Y-axis is the alarm rate and the X-axis is the network anomaly score threshold. It is observed that the normal vehicles generate a much lower alarm rate compared to abnormal ones. When threshold is 0.14, all abnormal vehicles are labeled positive by AnRAD while the false positive rate is at the order of 10e-2. Therefore, for vehicle anomaly detection tasks, the framework leaves a wide margin to trade between detection and false alarm.

AnRAD provides the reasoning ability, in that the positive labelings can be explained by introspecting the anomaly scores of the key lexicons. For instance, Table I shows the relationship between the key lexicon scores and the manually annotated anomaly classes. In this example, key lexicons generating an anomaly score higher than 0.8 are defined as *"outstanding"*. We count the outstanding occurrences and sort the top three lexicons for each annotated class. For speeding and sudden

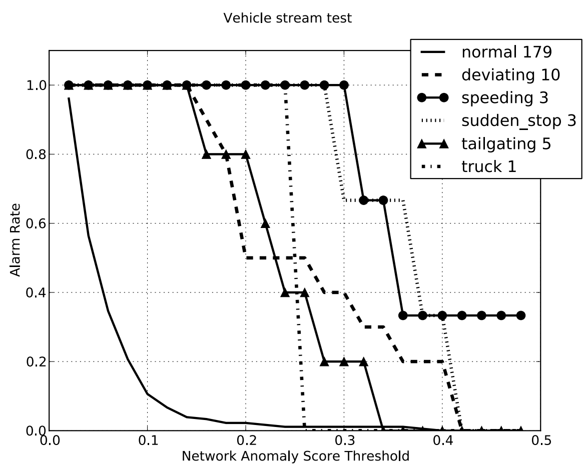


Fig. 4. Vehicle Detection Result

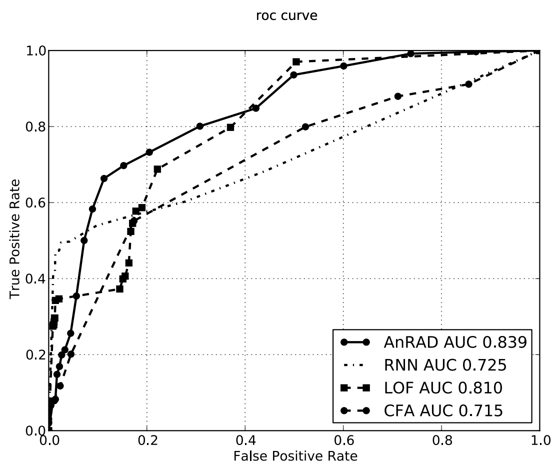


Fig. 5. DARPA dataset evaluation

TABLE I. CORRELATION BETWEEN ANOMALY TYPES AND OUTSTANDING NODES

Anomaly	Top 3 Outstanding Nodes
sudden stop	**1.** ⟨speed⟩; **2.** ⟨neighbor(1).speed⟩; **3.** ⟨neighbor(1).distance⟩
speeding sedan	**1.** ⟨speed⟩; **2.** ⟨neighbor(1).speed⟩; **3.** ⟨neighbor(1).distance⟩
tailgating	**1.** ⟨speed, neighbor(1).distance⟩; **2.** ⟨longitude, latitude, speed, direction⟩; **3.** ⟨longitude, speed, direction⟩
deviating from driveway	**1.** ⟨longitude⟩; **2.** ⟨longitude, latitude, neighbor(1).direction⟩; **3.** ⟨longitude, latitude, direction⟩
speeding truck	**1.** ⟨vehicle_size, longitude, longitude^{-2}⟩; **2.** ⟨latitude, speed, neighbor(1).distance, neighbor(1).speed⟩; **3.** ⟨vehicle_size, longitude, longitude^{-1}⟩

stops, such anomalies are closely related to vehicle speed; our analysis also shows that the most outstanding lexicon for this type of anomaly is ⟨speed⟩. Tailgating happens when one vehicle fast approaching another, so it can be explained by that the composite lexicon of speed and distance to the first neighbor has an increase in its anomaly score. Anomalies such as deviating from the road are usually coupled by high anomaly scores in coordinates-related lexicons. Finally, trucks may be caught speeding even if this speed was normal for a sedan. Such behavior causes high scores at the composite lexicons of vehicle size and the displacement in consecutive frames. The example shows that the AnRAD framework can provide explanation to its positive results without training labels or domain knowledge.

B. Comparative Evaluations

The proposed framework (AnRAD) is tested by fully labeled datasets and compared with other baseline methods. The baseline algorithms considered are: density-based method, incremental local outlier factor [20] (LOF); classification-based method, replicator neural network [13] (RNN); and rule-based method, cross-feature analysis [5] with CART decision trees (CFA). Although the baselines do not have all functions that AnRAD provides, such as reasoning and incremental training, here only the detection performances are considered.

The first dataset is processed from DARPA 1998 tcpdump files. For each IP address pairs, traffic statistics are recorded per 300ms-frame. In total, 21 primary features are extracted from the raw dump files. Some examples of the features are bytes from client (or server) to server (or client), service ports and number of clients connected with a server. We do not use the session-oriented KDD 99 dataset [26] because we would like to investigate concurrent data streams rather than session-oriented data points, and our processing also leverages less attack specific domain knowledge. The self-structuring network picks 123 key lexicons out of 446320 possibilities given $max\_order = 5$ for both feature-wise and temporal pooling. For training, normal streams from the seven weeks of training data are randomly sampled and about 20000 frames are selected. The negative class for test has another 7000 streams, and all the attacks (422 streams, 24 categories) in the seven weeks form the positive class. The moving window size is five frames for all methods.

The receiver operation curves (ROC) for comparing methods are plotted in Fig. 5 with X-axis representing the false alarm rate and Y-axis representing the true detection rate. Note that the true positive rates are averaged across anomaly categories to prevent the result from being biased by some large classes. The proposed method obtains the best area under curve (AUC) comparing to Incremental LOF, neural network and decision tree methods. So AnRAD has the advantage in the tradeoff between false alarm and detection rate. This is because the model construction is able to capture implicit patterns while the general baseline methods cannot. In this example, LOF and neural network methods outperform the decision tree approach because they work better with continuous features.

The second dataset is the system call sequences from the ADFA-LD dataset, which has discrete features. The training data consist of less than 20000 system calls. The testing data have about 6000 sequences from the validation set and 746 sequences from the attack set. To enable LOF, we use *Levenshtein Editing Distance*; for the neural network method, 100 frequent and orthogonal system calls are sampled from the training set as template points, and the input layer of the

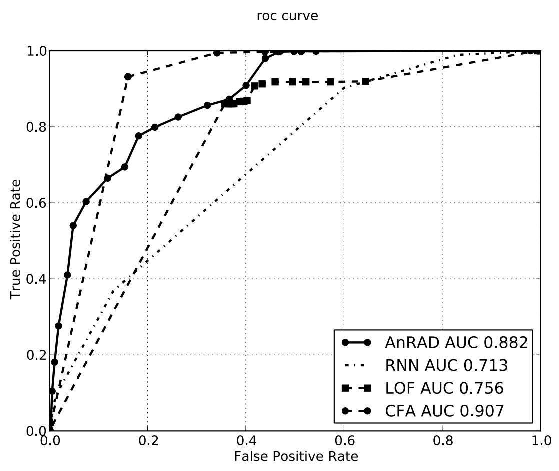


Fig. 6. ADFA-LD dataset evaluation

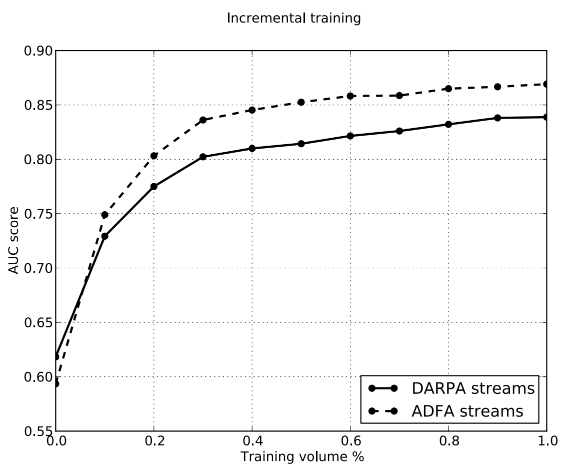


Fig. 7. AUC score change with training stream

network receives the calls' distances to these templates. The moving window size is set to six consecutive calls.

In Fig. 6, the AnRAD and the decision tree methods outperform the other two. This is because the latter two approaches cannot adapt to pure categorical features well. The decision tree has marginally better AUC score, but it suffers from the overfitting problem: it cannot reach a false positive rate lower than 19%, although the performance is good at high-detection-rate regions.

The comparisons in this section demonstrate the AnRAD framework's detection performance is competitive or superior to classical methods. Besides, AnRAD is the only method that provides incremental training, transparency and adaption to both continuous and categorical data.

C. Knowledge Base Refinement

Training in Section V-B uses clean samples, i.e. no anomalies are intentionally inserted. However in real unsupervised case, there is no guarantee of the training set quality. So it is important that the framework can incrementally improve the knowledge base quality. In this experiment, we firstly train confabulation networks with anomalous data. Then clean training sets are segmented into 10 episodes and fed into the knowledge base one by one. At each stage, the model tests the same evaluation set, and have the AUC scores collected.

In Fig. 7, for both DARPA and ADFA datasets, the score starts low because the anomalous training samples result in poor knowledge base quality. As more data stream received, the AnRAD framework is able to correct the erroneous knowledge continuously. We can observe that the detection performance being continuously optimized by new and better training samples. Therefore, the proposed episodic training method is effective.

VI. GPU Acceleration of Recall

Section V demonstrates the detection quality of AnRAD. However, deploying it for real-time detection requires computing performance optimization. Since the network structuring can be achieved offline and the training is already fast (less than $5ms/frame$ on the DARPA dataset) without optimization efforts, the bottleneck is the score calculation ($> 200ms/frame$ on the DARPA dataset) described in Section III. Fortunately, the confabulation network has inherent layered and massively parallel structure. And this can be exploited to improve the runtime performance of the recall process.

According to anomaly score equations (1) and (2), the AnRAD detection process must calculate the excitation level of each symbol in each key lexicon. The serial complexity of detecting one instance is $O(LDF)$, where L is the number of key lexicons, D is the average number of symbols in one key lexicon, and F is the average number of knowledge links connected to one key symbol. For parallel implementation at the node level, each key lexicon works as an independent test, so they can be distributed to multiple computing elements (i.e. CUDA blocks). In the symbol level, knowledge links can be distributed to different CUDA threads to enable parallel processing.

We propose to re-map the workload to the GPU as shown in Fig. 8. We assign a CUDA block for each key lexicon, or part of a large lexicon that consists of many candidate symbols. During the system initialization, the trained confabulation model is flattened and loaded to the GPU as an in-memory knowledge base. The input streams are organized into lexicon symbols and dynamically sent to the devices at each frame. A CUDA block computes the anomaly score of its key lexicon by accessing different portions of the knowledge base. Threads within a block utilize the shared memory to compute the anomaly score Equation (2) cooperatively.

The GPU recall implementation is evaluated by the three datasets in Table II. For serial implementation, programs run on Intel Xeon W5580 with 3.20GHz frequency; for GPU implementation, the device used is NVIDIA Tesla C2075 with 448 CUDA cores at 1.15GHz. Compared with the single-threaded CPU baseline, the parallel version reduces the runtime substantially. It is observed that at least a 200X speedup is achieved. The design fully exploits the concurrent structure of the confabulation model. Also, breaking down large lexicons alleviates the block resource requirement and helps the kernel achieve 100% theoretical occupancy.

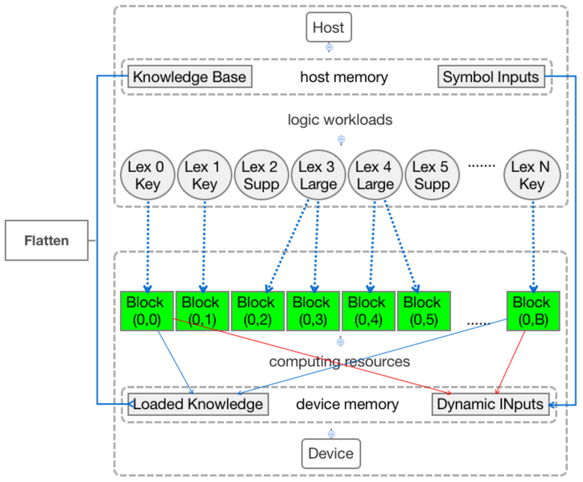


Fig. 8. Workload mapping on GPU

TABLE II. Acceleration of GPU Implementation

Dataset	Serial time	GPU time	Speedup
DARPA	200.5ms	0.375ms	535X
ADFA-LD	10.8ms	0.0402ms	269X
Vehicle	78.6ms	0.247ms	318X

VII. Conclusion

We have presented a self-structured confabulation framework that provides real-time anomaly detection for data streams. The framework learns the application-specific configuration of network hierarchy from the data. Results show competitive detection accuracy and reasoning ability without the aid of training labels. Furthermore, the recall algorithm is significantly accelerated by GPUs with fine-grained parallelization. In the future work, we will improve the workload distribution so that testing instance can be dynamically assigned to multiple heterogeneous devices.

Acknowledgment and Disclaimer

This work was supported by the Air Force Research Laboratory under contract FA8750-12-1-0251.

Received and approved for public release by AFRL on 03/06/2015, case number 88ABW-2015-0875. Any Opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of AFRL or its contractors.

References

[1] C. C. Aggarwal and S. Y. Philip. Outlier detection with uncertain data. In *SDM*, pages 483–493. SIAM, 2008.

[2] K. Ahmed, Q. Qiu, P. Malani, and M. Tamhankar. Accelerating pattern matching in neuromorphic text recognition system using intel xeon phi coprocessor. In *Neural Networks (IJCNN), 2014 International Joint Conference on*, pages 4272–4279. IEEE, 2014.

[3] E. Begoli. A short survey on the state of the art in architectures and platforms for large scale data analysis and knowledge discovery from data. In *Proceedings of the WICSA/ECSA 2012 Companion Volume*, pages 177–183. ACM, 2012.

[4] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: identifying density-based local outliers. In *ACM Sigmod Record*, volume 29, pages 93–104. ACM, 2000.

[5] J. B. Cabrera, C. Gutiérrez, and R. K. Mehra. Ensemble methods for anomaly detection and distributed intrusion detection in mobile ad-hoc networks. *Information Fusion*, 9(1):96–119, 2008.

[6] Q. Cai, H. He, and H. Man. Spatial outlier detection based on iterative self-organizing learning model. *Neurocomputing*, 117:161–172, 2013.

[7] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):15, 2009.

[8] Q. Chen, Q. Qiu, H. Li, and Q. Wu. A neuromorphic architecture for anomaly detection in autonomous large-area traffic monitoring. In *Computer-Aided Design (ICCAD), 2013 IEEE/ACM International Conference on*, pages 202–205. IEEE, 2013.

[9] Q. Chen, Q. Qiu, Q. Wu, M. Bishop, and M. Barnell. A confabulation model for abnormal vehicle events detection in wide-area traffic monitoring. In *Cognitive Methods in Situation Awareness and Decision Support (CogSIMA), 2014 IEEE International Inter-Disciplinary Conference on*, pages 216–222. IEEE, 2014.

[10] G. Creech and J. Hu. Generation of a new ids test dataset: Time to retire the kdd collection. In *Wireless Communications and Networking Conference (WCNC), 2013 IEEE*, pages 4487–4492. IEEE, 2013.

[11] G. Creech and J. Hu. A semantic approach to host-based intrusion detection systems using contiguous and discontiguous system call patterns. *IEEE transactions on pattern computer*, 63(4):807–819, 2013.

[12] Z. Fu, W. Hu, and T. Tan. Similarity based vehicle trajectory clustering and anomaly detection. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 2, pages II–602–5. IEEE, 2005.

[13] S. Hawkins, H. He, G. Williams, and R. Baxter. Outlier detection using replicator neural networks. In *Data Warehousing and Knowledge Discovery*, pages 170–180. Springer, 2002.

[14] R. Hecht-Nielsen. *Confabulation Theory: The Mechanism of Thought*. Springer Heidelberg, 2007.

[15] H. Huang. *Rank Based Anomaly Detection Algorithms*. PhD thesis, Syracuse University, NY, USA, 2013.

[16] I. Jolliffe. *Principal component analysis*. Wiley Online Library, 2005.

[17] B. Li, C. Wang, and D.-S. Huang. Supervised feature extraction based on orthogonal discriminant projection. *Neurocomputing*, 73(1):191–196, 2009.

[18] R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, et al. Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation. In *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings*, volume 2, pages 12–26. IEEE, 2000.

[19] P. Mitra, C. Murthy, and S. K. Pal. Unsupervised feature selection using feature similarity. *IEEE transactions on pattern analysis and machine intelligence*, 24(3):301–312, 2002.

[20] D. Pokrajac, A. Lazarevic, and L. J. Latecki. Incremental local outlier detection for data streams. In *Computational Intelligence and Data Mining, 2007. CIDM 2007. IEEE Symposium on*, pages 504–515. IEEE, 2007.

[21] Q. Qiu, Q. Wu, M. Bishop, R. E. Pino, and R. W. Linderman. A parallel neuromorphic text recognition system and its implementation on a heterogeneous high-performance computing cluster. *Computers, IEEE Transactions on*, 62(5):886–899, 2013.

[22] A. Roy. A theory of the brain-the brain uses both distributed and localist (symbolic) representation. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 215–221. IEEE, 2011.

[23] C. Sung, J. Woo, M. Goodman, T. Huffman, and Y. Choe. Scalable, incremental learning with mapreduce parallelization for cell detection in high-resolution 3d microscopy data. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–7. IEEE, 2013.

[24] A. S. Tanenbaum. Computer networks, 4-th edition, 2003.

[25] A. Zimek, R. J. Campello, and J. Sander. Ensembles for unsupervised outlier detection: challenges and research questions a position paper. *ACM SIGKDD Explorations Newsletter*, 15(1):11–22, 2014.

[26] Uci machine learning repository. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.