A Bayesian Approach To Distributed Anomaly Detection In Edge AI Networks

Murugaraj Odiathevar, Winston K.G. Seah, Marcus Frean
murugaraj.odiathevar, winston.seah, marcus.frean @ecs.vuw.ac.nz
School of Engineering and Computer Science
Victoria University of Wellington, New Zealand

***Abstract*—The challenge of anomaly detection is to obtain an accurate understanding of expected behaviour which is intensified when the data are distributed heterogeneously. Transmitting raw data to a central site incurs high communication overhead and raises privacy issues. The concept of Edge AI allows computation to be performed at the edge site allowing for quick decision making in mission critical scenarios such as self-driving cars. A model is learnt locally and its parameters are transmitted and aggregated. However, existing methods of aggregation do not account for variance and heterogeneous distribution of data. They also do not consider edge constraints such as limited computational, memory and communication capabilities of edge devices. In this work, a fully Bayesian approach is employed by means of a Bayesian Random Vector Functional Link AutoEncoder being incorporated with Expectation Propagation for distributed training. Our anomaly detection system operates without any transmission of raw data, is robust under inhomogeneous network densities and under uneven and biased data distributions. It allows for asynchronous updates to converge in a few iterations and is a relatively simple neural network addressing edge constraints without compromising on performance as compared to existing more complex models.**

***Index Terms*—Anomaly detection, Bayesian, Random Vector Functional Link, Expectation Propagation, Edge AI, Single Layer Feed-Forward Neural Network.**

I. Introduction

Anomaly detection is about finding patterns in data that do not conform to expected behaviour [1]. The challenge is to obtain an accurate understanding of expected behaviour. This is usually done by training a model using non-anomalous data. In the big data era however, gathering data poses a challenge because transmitting data through the network incurs high communication costs. In certain sensitive sectors such as healthcare, transmitting data also raises privacy concerns [2]. In applications such as smart cities, data are distributed and training and inference need to be made quickly such as in self-driving vehicles [3].

The shift in paradigm of pushing computational capabilities to the edge of the network can redefine anomaly detection systems, thus lowering communication costs [4], improves data privacy and avoids single point of failure. Anomaly detection, which depends on the application, plays a critical role in many fields such as surveillance and monitoring [5], Wireless Sensor Networks (WSNs) [6] and industrial IoT (Internet of Things) [7], [8]. For example, crash detection in traffic surveillance [9], invalid router updates in network connections [10], pedestrian detection in self-driving cars [11], anomalous condition detection in medical electrocardiograms [1], etc. are all mission critical scenarios which require quick decision making and tractability. Furthermore, privacy also matters. In this light, edge computing allows for quick training, decision making and preserves privacy. But learning locally at each site on local data are not sufficient to build a robust detection model especially when the data may vary and deep learning takes too much time, requires a lot of computation and the problem becomes intractable.

There have been proposals for distributed anomaly detection methods [8], [12]–[14] as well as algorithms to train a model in a distributed manner [15]–[19]. Unfortunately, devices found at the edge are not able to utilise advanced methods such as deep learning which are computationally heavy. The main research question is how can anomaly detection take into consideration heterogeneously distributed data given edge constraints? Edge constraints include limited computational, memory and network or communication capabilities of edge devices. The shortcomings of existing literature in addressing this is highlighted in section II. Hence, this work builds a robust, theoretically sound and relatively simple anomaly detection system in a distributed manner for edge devices without transmitting raw data over the network.

For anomaly detection, AutoEncoders (AE) have been extensively studied and used [20], [21]. However, optimising a deep neural network AE with distributed methods [18] has high computational complexities which is challenging in edge networks. In this work, we explore the Random Vector Functional Link (RVFL) network, which is simply a single hidden layer feedforward neural network (SLFN) [22]. Only one layer of weights need to be trained for RVFL which provides us with closed form solutions as it will be shown in section III. This allows us to incorporate a theoretically robust distributed training scheme which achieves the goal without the complexities of deep neural networks. In a RVFL, the weights between the input and the hidden layer are randomly initialised and fixed, and the hidden layer has a particularly high number of neurons as compared to the input layer. RVFLs have been used in many applications such as detecting wind power ramp [23], forecasting oil prices [24] and classification in data streams [25]. The simplicity of the RVFL architecture also allows for particularly rapid training.

Despite the benefits, not many methods have considered Bayesian approaches in the literature. Bayesian approaches allow the confidence or uncertainty of a model's output to be computed. Though they have more parameters to be trained, they provide a natural mechanism to cope with insufficient or poorly distributed data. In the Bayesian approach, the posterior distribution of the parameters is computed rather than a point estimate of the parameters. We will show that this allows for better aggregation as compared to other *ad*

hoc averaging methods. Scardapane *et al.* [22] implement Bayesian RVFL (BRVFL) for regression. In a separate work, they train a RVFL (non-Bayesian) in a distributed fashion [26] using two methods, by the averaging consensus method and by optimizing the global problem in a distributed manner to perform regression. In contrast, we adopt a full Bayesian approach by using Expectation Propagation (EP) [17] to train a BRVFL in a completely distributed manner for auto-encoding.

Much of the work in training a distributed model or a distributed anomaly detection system claim that they can perform under non-Independent and Identically Distributed (IID) data distributions but they are not thoroughly evaluated in the experiments. We take a step further to evaluate our method under biased and uneven data partitions.

In this paper, we introduce the Expectation Propagation Bayesian Random Vector Functional Link AutoEncoder (EP-BRVFL-AE). The contributions of this work are as follows:

(i) A fully Bayesian approach to build a distributed anomaly detection model addressing communication constraints and privacy issues.
(ii) An anomaly detection model which is robust under inhomogeneous network densities and under uneven and biased data distributions.
(iii) A relatively simple Neural Network addressing computation and memory constraints without compromising performance.

To the best of our knowledge, this is the first fully Bayesian-based distributed anomaly detection system built with BRVFL. It is also the first to evaluate the method under biased and uneven partitions of data explicitly. The paper is structured as follows. Section II presents recent related work. Section III describes the preliminaries. Section IV formulates the main algorithm. Section V explains the experiments conducted to evaluate the system and discusses results, and we conclude in section VI with a few directions for future work. [1]A schematic overview of the paper is shown in Figure 1 and an abbreviation list is given in Table I.

Table I: Nomenclatures of terms used in the manuscript

Abbreviation	Description
AE	AutoEncoders
BRVFL	Bayesian Random Vector Functional Link
SLFN	Single hidden Layer Feedforward Neural Network
EP	Expectation Propagation
FL	Federated Learning
VI	Variational Inference
IID	Idependent and Identically Distributed
ROC	Receiver Operating Characteristic
AUC	Area under ROC curve
PCA	Principal Component Analysis
KNN	K-Nearest Neighbours
LOF	Local Outlier Factor
GMM	Gaussian Mixture Model
OCSVM	One Class Support Vector Machine
ADPS	Average Degree Per Site
SGD	Stochastic Gradient Descent
MCMC	Markov Chain Monte Carlo

[1]This work does not raise any ethical issues

II. Related Work

The related work for the proposed EP-BRVFL-AE falls into three categories. Firstly, we will look at current methods of performing distributed training of neural networks. Secondly, we shall review methods which perform anomaly detection in a distributed manner. Lastly, Bayesian methods shall be reviewed to summarise the motivation behind the proposed algorithm.

A. Distributed training of models

The literature in this area consists mostly of averaging methods which have not considered different distributions and variance of data at each site. These scenarios are claimed but not accounted for in the evaluations. We explicitly evaluate our method under biased and uneven data partitions. Furthermore, these existing methods have high computational complexity, long training times and are not applicable for Bayesian approaches.

One well known method is the consensus algorithm [27] which is used to achieve agreement between sites, for example in blockchain technology. Neural networks can be trained using the consensus algorithm with decent results [28]. A privacy-preserving deep learning framework in IoT using random projection and differential privacy is presented in [7]. Other methods include a decentralized Average Consensus method and Alternating Direction Method of Multipliers [26].

Many other methods such as synchronous and asynchronous stochastic gradient descent (SGD), gradient accumulation, scatter-reduce-all gather method, binary blocks algorithm, etc. can also be used for neural network training [18], [29]. Communication patterns can also be optimised by using a parameter server and peer-to-peer communication patterns [30] or deep reinforcement learning for resource scheduling [31]. In the same light, a hierarchical method using max pooling, average pooling and concatenation of the parameters is used in [19]. Similarly, data communications is improved for distributed synchronous SGD by merging gradients of different layers [32]. As such, the trade off between local gradient descent updates and global aggregation is explored with respect to resource constraints to optimise communication in [33]. Furthermore, optimising SGD algorithms for communications only leads to slight a improvement and depending on the task and data, the gains may easily dissipate with increase in number of iterations required.

These distributed methods also fall under the term Federated Learning (FL) [34]. The key challenges of FL are that the training data are non-IID, unbalanced and massively distributed and communication is also limited [34], [35]. There are studies which provide frameworks for FL [36], [37] and tackle accuracy degradation in models due to imbalances in distributed training data [38]. A model similarity scheme to compare models trained on data in different locations and a protocol to classify data without transmitting model parameters helps preserve privacy but does not elaborate on distributed training [39]. Surveys of FL approaches such as FedAvg, FedProx, Local Gradient Descent and real-world

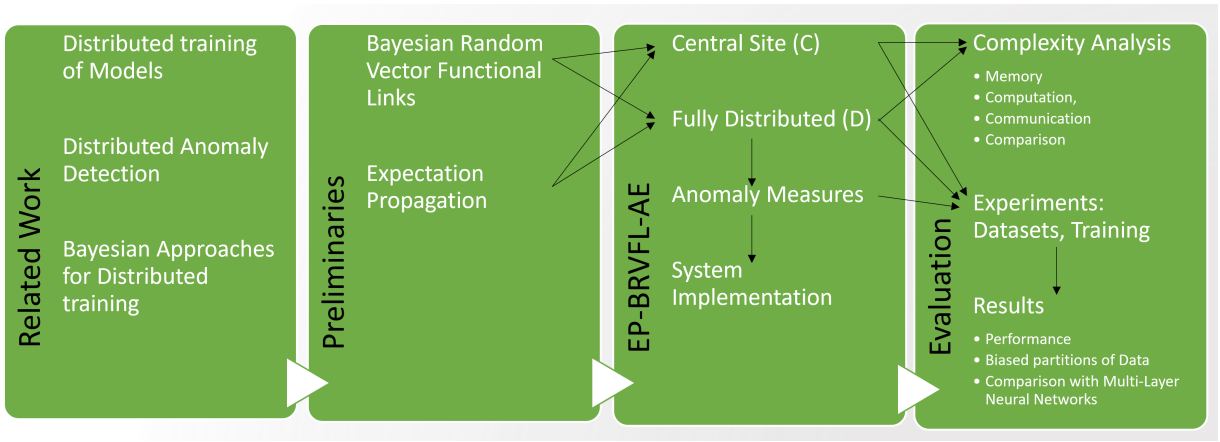


Figure 1: Schematic overview of the paper

challenges such as resource constraints, communication cost, storage and scheduling can be found in [40], [41].

Our method implicitly achieves FL with certain restrictions such as the use of SLFN to address learning in a resource constrained environment. Communication is reduced since convergence is achieved quickly as will be shown in Fig 8a. Moreover, our approach is also able to perform asynchronous updates. Unlike most of the existing work, we validate with an unevenly distributed data in section V-C3. This work is not a general FL method but shows that for specific application such as anomaly detection, FL can be achieved with the right model design and methodology.

B. Distributed Anomaly Detection

The methods in this area involve *ad hoc* averaging of parameters which may not be identifiable, or requires the assumption that the objective function is linearly separable which is not necessarily true. Furthermore, as noted previously, these methods do not account for uncertainty or variance during training and testing. Other methods of aggregation are problem specific and work simply on one-dimensional variables. Our fully Bayesian approach does not require the separability assumption and considers the variance and distribution of the data at each site. Some of the works also do not evaluate the scenario where the data are biased and unevenly distributed.

Breaking the global optimisation problem into a distributed one requires the assumption that the objective function is linearly separable. The equations of One Class Support Vector Machine (OCSVM) [12] and the Minimum Volume Elliptical Principal Component Analysis (MVE-PCA) [14] are reformulated into a decentralised optimisation function. Alternatively, an ensemble algorithm can be used to optimally weight different Random Forests models trained on heterogeneous data distributions by minimising uncertainty of predictions [13]. Similarly, Gaussian clusters trained on data from the edge can be merged at the fog and the cloud levels [8]. In these, only the summary of the data or parameters necessary for model training are being transmitted. Another approach to distributed training involves partitioning the data space. This is done by Local Outlier Factors (LOF) eliminating certain points from memory [42].

By analyzing the topology and installing rules to collect statistics at optimal monitoring positions, forwarding anomalies are detected and located in the network [10]. For a prediction variance anomaly detector, the most vital component is the covariance matrix. Instead of collecting all data segments centrally, the matrix is aggregated using compressed difference sequences and the sample standard deviation at each site for anomaly detection in WSNs [6].

Weights of Gated Recurrent Units trained at different sites are averaged based on number of data points for anomaly detection [43]. FL is used to train a deep Long Short Term Memory model for anomaly detection with a focus on communication efficiency through a gradient compression mechanism [44]. These models are employed only on one dimensional sequential data. A Multi-task Deep Neural Network is trained using FL in a supervised manner for anomaly detection and traffic classification in network data [45]. Furthermore, as mentioned previously, training deep learning models at the edge is challenging computationally and memory-wise. A multi-layer neural network is evaluated in section V-D. Furthermore, these recent works do not consider variance in the data or perform Bayesian averaging which provides more information on the data as explained in the following subsection.

C. Bayesian approaches for distributed training

Compared to the methods discussed thus far, Bayesian methods provide a distribution over the parameters of the model instead of a point estimate. This allows for a richer representation and the uncertainty of a model's output to be computed. It allows variance to be considered when combining or averaging model parameters to achieve a global model. They are also effective when there is not enough data. However, it increases the number of parameters within the model. Hence, using deep learning methods with a myriad of parameters in a Bayesian way is already a challenge in the cloud. But with a RVFL, we can reduce the complexity substantially to allow for efficient training at the edge.

Parallel Markov Chain Monte Carlo (MCMC) is performed to obtain a "subposterior" in each site using a "fractional" prior and then combining them [15]. The "fractional" prior may be too weak to effectively regularize such as when the likelihood

	Methods	Shortcomings
Distributed Training	Average Consensus [26]–[28], [33]	Heterogeneously distributed data or variance of data at each site is not considered.
	Optimising SGD [18], [19], [29]–[32]	Heterogeneously distributed data or variance of data at each site is not considered, intractable computation and high communication and computational complexity.
Distributed Anomaly Detection	Reformulation of optimisation function [12], [14], [42]	Heterogeneously distributed data or variance of data at each site is not considered and assumes objective function is linearly separable.
	Ensemble methods [6], [8], [10], [13]	Heterogeneously distributed data or variance of data at each site is not considered
	Deep Neural Networks [43]–[45]	Intractable computation and high computational and memory complexity for edge sites.
Bayesian Approaches	EP and/or MCMC [15], [17], [46]	High computational complexity for edge sites.

Table II: Summary of related work

is not sufficient for a good approximation of the posterior [17]. Another approach would be to multiplicatively combine the posteriors of each site and divide the result by $K-1$ priors, where K is the number of sites. This however runs into computational instabilities [17]. Expectation Propagation (EP) and MCMC is used in [46] but their method achieves slow convergence and has more variability in the beginning [17]. Xu *et al.* [47] also use EP and MCMC to perform inference. Though MCMC guarantees convergence in the limit, it is not easy to determine the number of steps required [48].

A summary of the related work and the shortcomings is found in Table II. We summarise our motivation as follows. Firstly, a Bayesian model provides us with uncertainty estimates which improves anomaly detection. Secondly, rapid training and testing is achieved with a SLFN such as the RVFL. Having to determine only one layer of weights is computationally tractable and gives us a closed form solution for the posterior. Speed and tractability are important aspects for anomaly detection and remediation, especially at the edge. Next, AE provides us with reconstruction error for anomaly scoring. Last but not least, EP can be incorporated to the Bayesian model to achieve distributed training of the model. The summary of each of the components and how they correlate are shown in Fig 2.

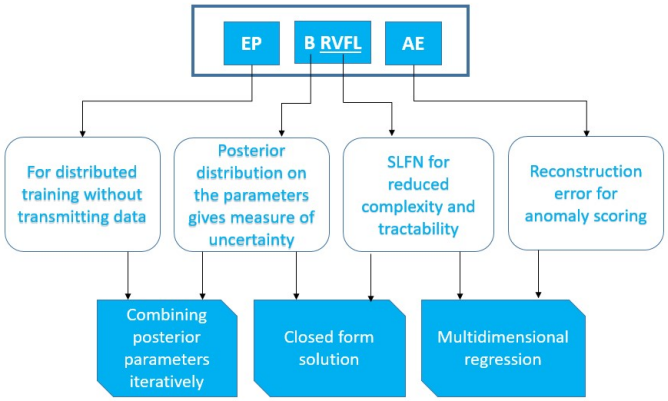


Figure 2: Summary of EP-BRVFL-AE and its components.

III. Preliminaries

We briefly introduce the Bayesian neural net we used at edge sites, and the Expectation Propagation (EP) algorithm used to combine information.

A. Bayesian Random Vector Functional Links

Let $\mathbf{x} \in \mathbb{R}^d$ be the input vector. The output of a neural network with one hidden layer is denoted by

$$f(\mathbf{x}) = \sum_{l=1}^{B} w_l h_l(\mathbf{x}) = \mathbf{w}^T \mathbf{h}(\mathbf{x}) \tag{1}$$

where $h_l(\cdot)$s are randomly initialised non-linear maps which project the input to a higher B dimensional space. We use the classical sigmoid function with weights $\mathbf{a} \in \mathbb{R}^B$ and biases $b \in \mathbb{R}$ drawn from a uniform distribution in the range $[-\lambda, \lambda]$ where λ is a positive real number as shown [22].

$$h_l(\mathbf{x}) = \frac{1}{1+\exp(-(\mathbf{a}^T\mathbf{x}+b))} \tag{2}$$

Structuring the above problem as a ridge regression problem with dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)|i = 1 \dots n\}$ for $y_i \in \mathbb{R}$, the optimal weights $\mathbf{w}$ are found by solving

$$\mathbf{w}^* = \underset{\mathbf{w}\in\mathbb{R}^B}{\operatorname{argmin}} \left\{ \frac{1}{2}\|\mathbf{H}\mathbf{w} - \mathbf{y}\|^2 + \frac{C}{2}\|\mathbf{w}\|^2 \right\}. \tag{3}$$

In equation (3), $\mathbf{H}$ is built by stacking row-wise vectors $\mathbf{h}(\mathbf{x}_i)$ and C is the regularisation factor. The solution can be found analytically via the Moore-Penrose inverse:

$$\mathbf{w}^* = (\mathbf{H}^T\mathbf{H} + C\mathbf{I})^{-1}\mathbf{H}^T\mathbf{y}. \tag{4}$$

To perform autoencoding, we consider a multidimensional output instead, replacing y_i by $\mathbf{y}_i$, where each dimension is independent. We then obtain a multidimensional regression problem where the equations are as before but $\mathbf{w}$ is now $B \times d$ and is independent along each dimension $j = 1, \dots, d$. This independence formulation is possible as we are only training one set of weights.

Let X denote the data. $P(X|\mathbf{w})$ denotes the likelihood and $P(\mathbf{w})$ denotes the prior. The posterior distribution over the parameters is given by Bayes law:

$$P(\mathbf{w}|X) \propto P(X|\mathbf{w})P(\mathbf{w}). \tag{5}$$

We use the basic results from [22], [49] for Bayesian ridge regression. We allow each of the outputs to vary by white noise with variance σ^2, resulting in (6) for the likelihood in each dimension. $\mathcal{N}(x|a, b)$ denotes a Gaussian distribution over x with mean a and variance b. For the prior, we incorporate our belief that the weights will be close to zero and express it as a zero-mean multivariate Gaussian with diagonal covariance.

In equation (7), $\mathbf{m}_0 = \mathbf{0}$ and $\mathbf{S}_0 = \gamma^{-1}\mathbf{I}$ for all j and γ is the precision parameter.

$$P(X|\mathbf{w}_j, \sigma^2) = \mathcal{N}(x_{ij}|\mathbf{w}_j^T\mathbf{h}(\mathbf{x}_i), \sigma^2) \tag{6}$$

$$P(\mathbf{w}_j|\mathbf{m}_0, \mathbf{S}_0) = \mathcal{N}(\mathbf{w}_j|\mathbf{m}_0, \mathbf{S}_0) \tag{7}$$

The posterior is again Gaussian, with mean $\mathbf{m}_j$ and covariance $\mathbf{\Sigma}$ given by

$$\mathbf{m}_j = \mathbf{\Sigma}(\mathbf{S}_0^{-1}\mathbf{m}_0 + \frac{1}{\sigma^2}\mathbf{H}^T\mathbf{x}_j) \tag{8}$$

$$\mathbf{\Sigma} = (\mathbf{S}_0^{-1} + \frac{1}{\sigma^2}\mathbf{H}^T\mathbf{H})^{-1} \tag{9}$$

If we take σ^2 to be constant for each j then $\mathbf{\Sigma}$ depends only on the variation in the data given by $\mathbf{H}^T\mathbf{H}$. The predictive distribution is also Gaussian,

$$P(\hat{\mathbf{x}}_j|\hat{\mathbf{x}}, \sigma^2, \gamma) = \mathcal{N}(\hat{\mathbf{x}}_j|\mathbf{m}_j^T\mathbf{h}(\hat{\mathbf{x}}), \phi(\hat{\mathbf{x}})^2) \tag{10}$$

$$\phi(\hat{\mathbf{x}})^2 = \sigma^2 + \mathbf{h}(\hat{\mathbf{x}})^T\mathbf{\Sigma}\mathbf{h}(\hat{\mathbf{x}}). \tag{11}$$

The method above provides us not only with an estimate for the mean of the weights, but also a variance for the weights and therefore, a predictive variance for $\mathbf{y}$. Furthermore, the solution is in closed form. Instead of specifying values for the hyper-parameters σ^2 and γ, we place a conjugate prior distribution on them, perform Bayesian inference and obtain Maximum-A-Posterior (MAP) values from the data.

B. Expectation Propagation

Expectation Propagation (EP) was formally introduced by Opper and Winthler [50] and generalised by Minka [51]. It is an iterative message-passing algorithm which minimises the Kullback-Leibler (KL) divergence of the reverse form $\mathrm{KL}[P(\mathbf{w}|X)||q(\mathbf{w}|\theta)]$ [49] between two distributions. From (5), if there is no closed form solution, Variational Inference (VI) [49] can be used to estimate the posterior. It is then projected on to q, a member of the family of exponential distributions, by matching its moments [49], [52]. There is no guarantee of convergence but it has been shown to work well for models with log-concave factors such as the Gaussian distribution [17]. In the case where the posterior is in the exponential family, minimizing the divergence conveniently corresponds to matching the moments [51].

As in FL, if a global model exists, the IID assumption is invoked with respect to the global model for the data, though they may be unevenly balanced between sites. This may also be viewd as performing Bayesian inference site by site taking the posterior as the new prior in the subsequent iteration. Else, other alternatives such as learning distinct local models should be considered [35]. From (5), the likelihood is factored into partitions, one for each site in the network. These are then combined iteratively with an approximate prior to produce the global posterior upon convergence. Let there be $k = 1, \ldots, K$ sites and X_k denote the data in the k^{th} edge site. We omit the dimension variable, j, to reduce clutter.

$$P(\mathbf{w}|X) \propto \prod_{k=1}^{K} P(X_k|\mathbf{w})P(\mathbf{w}) \tag{12}$$

is approximated by

$$g(\mathbf{w}|\mathbf{r}, \mathbf{Q}) \propto \prod_{k=1}^{K} g_k(\mathbf{w}|\mathbf{r}_k, \mathbf{Q}_k)g_0(\mathbf{w}|\mathbf{r}_0, \mathbf{Q}_0), \tag{13}$$

where $g(w|\mathbf{r}, \mathbf{Q})$ is a member of the exponential family and $\mathbf{r}, \mathbf{Q}$ are the natural parameters. The exponential family of distributions is closed under multiplication and it translates to addition of the natural parameters. The EP algorithm is stated as follows. Firstly, initialise $\mathbf{r}_k, \mathbf{Q}_k = 0$ and $\mathbf{r}_0, \mathbf{Q}_0$ to a global prior. This global prior is the natural parameters of the prior in (7). Then $\mathbf{r} = \mathbf{r}_0 + \sum_{k=1}^{K}\mathbf{r}_k$ and $\mathbf{Q} = \mathbf{Q}_0 + \sum_{k=1}^{K}\mathbf{Q}_k$.

E1: At each site, determine the cavity distribution g_{-k}, by $\mathbf{r}_{-k} = \mathbf{r} - \mathbf{r}_k$, $\mathbf{Q}_{-k} = \mathbf{Q} - \mathbf{Q}_k$.

E2: At each site, approximate the tilted distribution $g_{\backslash k}$ where $g_{\backslash k}(\mathbf{w}) \propto P(X_k|\mathbf{w})g_{-k}(\mathbf{w})$, using VI [49] if no closed form solution is available, and by matching moments.

E3: At each site, compute the change in distribution. $\Delta\mathbf{r}_k = \mathbf{r}_{\backslash k} - \mathbf{r}_{-k} - \mathbf{r}_k$ and $\Delta\mathbf{Q}_k = \mathbf{Q}_{\backslash k} - \mathbf{Q}_{-k} - \mathbf{Q}_k$

E4: At each site, update the distribution with a damping factor $\delta \in (0, 1]$, $\mathbf{r}_k \leftarrow \mathbf{r}_k + \delta\Delta\mathbf{r}_k$ and $\mathbf{Q}_k \leftarrow \mathbf{Q}_k + \delta\Delta\mathbf{Q}_k$

E5: In a central site, update the global parameters, $\mathbf{r} \leftarrow \mathbf{r} + \delta\sum_{k=1}^{K}\Delta\mathbf{r}_k$ and $\mathbf{Q} \leftarrow \mathbf{Q} + \delta\sum_{k=1}^{K}\Delta\mathbf{Q}_k$

Repeat steps **E1**-**E5** until $\Delta\mathbf{r}_k$ and $\Delta\mathbf{Q}_k$ are small or when the tilted distribution at each site is consistent with the approximate posterior. For further details, we refer readers to [17]. Assuming each of the weights of the BRVFL-AE follows a Gaussian distribution as in [22], EP can be used in closed form to compute the tilted distribution. There is a one-to-one mapping between the natural parameters and the mean and covariance of the Gaussian distribution:

$$\mathbf{r} = \mathbf{\Sigma}^{-1}\mathbf{m} \qquad \mathbf{Q} = \mathbf{\Sigma}^{-1} \tag{14}$$

IV. EP-BRVFL-AE

The main method is firstly formulated for the case with a central site, and then extended to a completely distributed version. The hyper-parameters are estimated likewise. Subsequently, the measures to perform anomaly detection are described, and a system diagram is given for implementation.

A. Central Site

The formulation of EP-BRVFL-AE follows naturally from the methods in Section III. An AE consists of an encoder network, a latent layer and a decoder network. The encoder maps the input data into the latent layer and the decoder reconstructs them. For the RVFL, the encoder network is fixed and maps the input data into a higher dimension. We do not implement skip connections from input to output, as when autoencoding through a SLFN, the model could merely learn the identity function.

Algorithm 1 presents EP-BRVFL-AE(C) with a central site. Site parameters, $\mathbf{r}_k, \mathbf{Q}_k$ are initialised to zero and global parameters $\mathbf{r}$ and $\mathbf{Q}$ are initialised to $\mathbf{0}$ and $\gamma\mathbf{I}$ respectively. The central site holds in memory a copy of the edge sites' parameters as well, to determine the best δ with respect to all sites. In **E2** of each iteration of EP, the prior is the cavity distribution. It is necessary that $|\mathbf{Q}_{-k}| > 0$. Else, the

Algorithm 1 EP-BRVFL-AE(C)

1: Initialise site and global parameters, $\mathbf{r}_k, \mathbf{Q}_k, \mathbf{r}, \mathbf{Q}$

2: **while** $\Delta\mathbf{r}_k$ and $\Delta\mathbf{Q}_k$ are large **do**
At each site, k: ▷ Site Operation
3: Update site distribution as in **E4** ▷ 2^{nd} iter. onw.
4: Determine cavity distribution as in **E1**
5: Compute the tilted distribution with (8) and (9) using cavity as prior
6: Compute $\Delta\mathbf{r}_k$ and $\Delta\mathbf{Q}_k$ as in **E3**

7: Send $\Delta\mathbf{r}_k$ and $\Delta\mathbf{Q}_k$ to central

At Central: ▷ Central Operation
8: **for** $k = 1, \ldots, K$ **do**
9: Initialise $\mathbf{Q}_{-k} = 0$, $\delta = 2\delta_0$.
10: **while** $|\mathbf{Q}_{-k}| \leq 0$ **do**
11: $\delta \leftarrow \delta/2$
12: **for** $k = 1, \ldots, K$ **do**
13: Calculate $\mathbf{r}_k, \mathbf{Q}_k$ as in **E4**
14: **end for**
15: Calculate $\mathbf{r}, \mathbf{Q}$ as in **E5**
16: Calculate cavity parameter $\mathbf{Q}_{-k}$ as in **E1**
17: **end while**
18: **end for**

19: Send $\delta, \mathbf{r}, \mathbf{Q}$ to sites
20: **end while**

Algorithm 2 EP-BRVFL-AE(D) without Central Site

1: Initialise site and global parameters, $\mathbf{r}_k, \mathbf{Q}_k, \mathbf{r}, \mathbf{Q}$

2: **while** $\Delta\mathbf{r}_k$ and $\Delta\mathbf{Q}_k$ are large **do**
At each site, k: ▷ Site Operation
3: Determine cavity distribution as in **E1**
4: Compute the tilted distribution with (8) and (9) using cavity as prior
5: Compute $\Delta\mathbf{r}_k$ and $\Delta\mathbf{Q}_k$ as in **E3**

6: Broadcast $\Delta\mathbf{r}_k$ and $\Delta\mathbf{Q}_k$
7: Receive $\Delta\mathbf{r}_k$ and $\Delta\mathbf{Q}_k$ from other sites

8: Initialise $\mathbf{Q}_{-k} = 0$, $\delta = 2\delta_0$.
9: **while** $|\mathbf{Q}_{-k}| \leq 0$ **do** ▷ Central Operation
10: $\delta \leftarrow \delta/2$
11: Calculate $\mathbf{r}_k, \mathbf{Q}_k$ as in **E4**
12: Calculate $\mathbf{r}, \mathbf{Q}$ as in **E5**
13: Calculate cavity parameter $\mathbf{Q}_{-k}$ as in **E1**
14: **end while**
15: **end while**

damping factor δ needs to be reduced. When the data are evenly distributed, the value of δ is rarely reduced. As the BRVFL method performs closed-form updates, the algorithm generally converges in two or three iterations. Our experiments also verify both these aspects.

The computation of update, cavity and tilted distribution (and changes) shall be referred to as "Site operation". The update computation at the site begins on the second iteration (line 3). The computation to aggregate the changes and to determine δ shall be referred to as "Central operation".

B. Fully Distributed

Algorithm 2 gives a fully distributed version, EP-BRVFL-AE(D). Each edge site holds a copy of the global parameters and incorporates new information when it is received. The central operation will be performed at the edge site. The updates from each site is broadcast throughout the network.

Figure 3: A network with 5 sites is shown. The left image is distributed with $\kappa = 1$ and the right image has a central site.

Firstly, we define a few terms to describe network topology. Network density κ is defined in (15). The Average Degree Per Site (ADPS) in (16) indicates the average number of neighbours per site. E is the total number of edges in the network and K is the total number of sites. A fully connected network has $\kappa = 1$ while a ring network has $\kappa = 2/(K-1)$. The Maximum Number of Hops required for any site to communicate in the network is denoted as MHs.

$$\kappa = \frac{2E}{K(K-1)} \tag{15}$$

$$\text{ADPS} = \frac{2E}{K} \tag{16}$$

If a site receives updates $\Delta\mathbf{r}_k$ and $\Delta\mathbf{Q}_k$ from all other sites, it will achieve the same result as having a central site. In other words, EP-BRVFL-AE(C) and EP-BRVFL-AE(D) with $\kappa = 1$ are equivalent if all other parameters are kept constant. An example diagram is shown in Fig 3. For networks with $\kappa < 1$, waiting for MHs steps to occur before performing an update is ideal, but the training can also proceed with updates as they arrive. In the latter case, an update from an edge site which is 3 hops away will only be incorporated on the third iteration. The clear advantage of not waiting is that there is no need for the site to have any knowledge of the network topology. Table III shows an example of a network and how two of the sites are updated. Since site A is 3 hops away from site E, the first update from site A will reach site E on the 3rd iteration.

	Updates received and incorporated	
Iter.	**At site C**	**At site E**
2	A1, B1, D1	D1, F1
3	A2, B2, D2, E1	D2, F2, C1
4	A3, B3, D3, E2, F1	D3, F3, C2, B1, A1
⋮	⋮	⋮

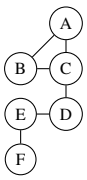


Table III: Numbers represent iterations and letters represent sites. C1 is the update that is broadcast from site C after the first iteration. C2 consists of updates from A1, B1, D1, and its own data.

Algorithm 2 appears simpler and with less computation per iteration but it requires more communication to reach

convergence. This reduction in computation is mainly due to each site being interested only in the δ value that works for itself. Our experiments showed that the first update from each site is the most important. This is because we are only transmitting the changes and in the first iteration, the flat initial prior allows the data to provide the most significant changes.

C. Anomaly measures

After Algorithm 1 reaches convergence, the global parameters $\mathbf{r}, \mathbf{Q}$ are used as the parameters for the BRVFL-AE in each site. After Algorithm 2 reaches convergence, each site's global parameters would have converged to a similar value. Anomaly scoring can be performed using one of three measures.

The common scoring method is using the reconstruction error (RE) for AE. The MAP estimate can be used for the weights. The predictive variance, $\phi(\hat{\mathbf{x}})^2$ can also be used as the measure or the confidence score for RE. Areas with high variance suggest that there is not enough data in the neighbourhood and thus, the point is more anomalous. A heuristic, $\mathcal{H}$, being a combination of both measures is also evaluated in this paper.

$$RE = \sum_{j=1}^{d} \|\mathbf{m}_j^T \mathbf{h}(\hat{\mathbf{x}}) - \hat{\mathbf{x}}_j\|^2 \tag{17}$$

$$\mathcal{H} = RE \times \phi(\hat{\mathbf{x}}) \tag{18}$$

D. System Implementation

A block diagram for EP-BRVFL-AE(D) at the edge site is depicted in Fig 4. After convergence, the data used to train the anomaly detection model can be discarded to save memory. The method can be adapted to perform online batch training with new data as well. Furthermore, in the event of transmission delay, asynchronous updates are also possible. We will show in section V-C3 that the optimal waiting time is equivalent to the maximum hop time for any two sites in the network to communicate their updates, and a delayed site update included in the next iteration would still converge to the same anomaly detection model.

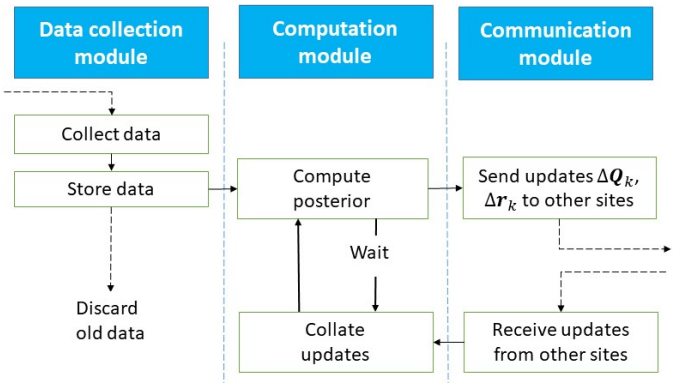


Figure 4: Block Diagram of EP-BRVFL-AE(D) implementation on edge site.

V. Evaluation and Discussion

In this section, we evaluate the algorithm's effectiveness with various datasets, analyse its computational, memory and communication complexity and compare its performance with a few other methods. In section V-C3, we explicitly split the dataset into biased and uneven partitions to showcase the efficacy of Bayesian averaging on our model. Other studies on distributed training have not evaluated their methods in this manner.

A. Complexity Analysis

Let c denote the transmission cost of network transmission per parameter, t denote the number of iterations required to obtain δ and s be the number of EP iterations needed for convergence. For the completely distributed setting, let e_k be the number of neighbours for site k.

1) Memory: Each site firstly holds n_k data points of dimension d. The parameters $\mathbf{r}$ and $\mathbf{Q}$ contain Bd and $B(B+1)/2$ quantities respectively. In EP-BRVFL-AE(C), at each site, the natural parameters $\mathbf{r}_k, \mathbf{Q}_k$ and the changes $\Delta\mathbf{r}_k, \Delta\mathbf{Q}_k$ are stored. At the central site, the global parameters $\mathbf{r}, \mathbf{Q}$, parameters of all of the sites and the sum of changes of the sites are stored. For EP-BRVFL-AE(D), $4(Bd+B(B+1)/2)$ quantities are stored at each site adding the global parameters and the sum of changes from neighbouring sites. Hence, memory is dominated by $\mathcal{O}(B^2)$.

2) Computation: For both algorithms 1 and 2, the cost of computing $\mathbf{H}$ at each site is $n_k(Bd+1)$. At each site, the update, cavity and change computation involving addition or subtraction costs $Bd + B^2$ each. For the tilted distribution, computing $\mathbf{H}^T\mathbf{H}$, $\mathbf{H}^T\mathbf{X}_k$ and $\mathbf{m}$ and inverting $\mathbf{\Sigma}$, cost $B^2 n_k$, $Bn_k d$, B^2 and B^3 respectively. If $B > n_k$ then the computation cost is dominated by $\mathcal{O}(B^3)$ else, it is dominated by $\mathcal{O}(B^2 n_k)$.

For EP-BRVFL-AE(C), summing up the updates costs $K(Bd + B^2)$ at the central site. To compute δ, an additional cost of $2K(Bd + B^2)$ for the update and cavity computation and B^3 to obtain the determinant is required. This computation is repeated t times. For EP-BRVFL-AE(D), summing up the updates is bounded above by $K(Bd + B^2)$, the update and cavity computation costs $2(Bd + B^2)$ and the determinant computation costs B^3. This is repeated t times until δ is determined. In our experiments, if the data are evenly distributed the initial value of $\delta_0 = 1$ suffices.

3) Communication: The sites transmit the changes in the natural parameters. In EP-BRVFL-AE(C), the central site transmits the global natural parameters with the same cost, $cK(Bd+B(B+1)/2)$ to all sites. To transmit δ, the cost is cK. In EP-BRVFL-AE(D), the edge sites broadcast the changes to the network. Depending on κ, the communication at each iteration has an upper bound of $cK(Bd + B(B+1)/2)$.

The total cost is total computation and communication at both edge and central times s. We will show that convergence is achieved with $s = 2$ for EP-BRVFL-AE(C) and $s =$Max Hops for EP-BRVFL-AE(D). Considering only the dominating terms, the total computational and communication complexity is $\mathcal{O}(s(cKB^2 + \max_k(n_k B^2, B^3)))$. Amount of data, n_k

is under computation and not under communication. With big data, $n_k >> B^2$, this presents significant savings in communication complexity of the network.

4) Comparison: EP-BRVFL-AE has a lower order of complexity or comparable to other methods. It depends on the number of data items, n_k, linearly, while B is a fixed parameter. MVE-PCA is cubic in n_k. Table IV gives a summary of these complexities.

	Memory	**Communication**	**Computation**
Edge (C)	$\mathcal{O}(B^2 + n_k d)$	$\mathcal{O}(cB^2)$	$\mathcal{O}(B^3 + n_k B^2)$
Central (C)	$\mathcal{O}(KB^2)$	$\mathcal{O}(cKB^2)$	$\mathcal{O}(tB^3)$
Edge (D)	$\mathcal{O}(B^2 + n_k d)$	$\mathcal{O}(cKB^2)$	$\mathcal{O}(tB^3 + n_k B^2)$
MVE-PCA [14]	$\mathcal{O}(d^2 + n_k d)$		$\mathcal{O}(n_k^3)$
Ellipsoidal[a] Clustering [8]	$\mathcal{O}(\rho_k d^2)$		$\mathcal{O}(\rho_k^2 p + n_k + \rho_k d^3)$

[a] p and ρ_k is the number of nearest ellipsoidals and number of ellipsoidals at each site respectively

Table IV: Complexity comparisons: Edge (C) and Edge (D) denote the edge sites for EP-BRVFL-AE(C) and EP-BRVFL-AE(D) respectively.

B. Experiments

1) Datasets: We evaluate our method on various datasets. Firstly, we select the UNSW-NB15 dataset (UNSW) because it has a hybrid of real modern normal and contemporary synthesized attack [53]. Secondly, we use the NSL-KDD [54]. We also use other real-world datasets from the UCI machine learning repository [55]. In the datasets, the class with more data samples is considered normal class and the others are combined to form the anomaly class. Four datasets from different applications are used, namely, Abalone, PageBlocks, Shuttle and Australian Credit Approval. The datasets are randomly partitioned to achieve a balanced set for testing. The balanced set is important for AUC measure (Section V-B3) to indicate the performance of the model effectively. The details are found in Table V.

Table V: Datasets Summary

Dataset	**Application Domain**	**Attributes**	**No. Instances Training (Testing) (Normal+Anomaly)**
UNSW-NB15	Intrusion Detection	42	56000 (37000+45332)
NSLKDD 2009	Intrusion Detection	41	67340 (9711+12833)
Abalone	Biology	8	1880 (94+94)
PageBlocks	Text Recognition	10	4353 (560+560)
Shuttle	Sensor Monitoring	9	34108 (3022+3022)
Australian Credit Approval	Finance	14	283 (100+100)

Algorithm 3 Preprocessing

1: **for** $k = 1, \ldots, K$ **do**
2: **At edges**: Send $\bar{f}_j^k$ to central
3: **end for**
4: **At central**: $\bar{f}_j = \frac{1}{K}\sum_{k=1}^{K} \bar{f}_j^k$
5: **At central**: Compute l_j and send to edges
6: **At edges**: Normalize data using (19)

2) Training: Firstly, before any training can occur, data from all of the sites need to be normalised appropriately. The method used by Su *et al.* [56], given by

$$\text{Normalization of } f_{ij} = \frac{1 - e^{-l_j f_{ij}}}{1 + e^{-l_j f_{ij}}}. \tag{19}$$

is used. In this equation, f_{ij} is the i^{th} observed value of feature j, e represents the Euler number ($e \approx 2.718$) and l_j is a constant. Since we want to adapt to normal traffic, the constant l_i is determined such that the average of feature j of the training data instances is mapped to 0.5. For this purpose, in EP-BRVFL-AE(C), the mean of each numerical feature, $\bar{f}_j^k$ in the data from each site k is transmitted to the central site. The global mean of each of those features is computed and l_j is determined and sent back to the edge sites for normalisation.

In EP-BRVFL-AE(D), a broadcast of $\bar{f}_j^k$ to all sites is required. This incurs an additional communication cost of d and computation cost of Kn_k. Alternatively, each site can use its own data to perform normalisation especially if the data distribution is similar between the sites.

The middle layer of the BRVFL-AE shall be ζ times the input layer d, and $\zeta = 10$ for comparison unless otherwise mentioned. We implement the algorithm with $\delta_0 = 1$. The number of edge sites is fixed to 10 for comparison and the data are randomly partitioned unless specified. The hyper-parameters are the MAP estimates of the hyper-posteriors determined by the EP. The initial hyper-prior parameters are such that the initial estimates are $\sigma^2 = 0.01$ and $\gamma = 0.01$ as in [22].

All experiments are implemented using Python 3.8.2 and run on GNU/Linux x86_64 with an Intel Core i7-7567U CPU at 3.5 GHz, 16 GB RAM.

3) Evaluation Measures: The methods are evaluated based on Receiver Operating Characteristic (ROC) curve and Area Under ROC Curve (AUC). The ROC curve is the plot of True Positive Rate (TPR) against False Positive Rate (FPR) at different threshold settings. The AUC is a one-value measure of the ROC curve quality and it represents the degree of separability between anomalies and normal traffic. A perfect model has an AUC value of 1, while a value of 0.5 or less suggests that the model has no capacity to separate the classes. The AUC scores are multiplied by 100 for display.

To evaluate convergence, we introduce two other measures. For the global parameters, relative difference is used to measure convergence [14]. Let $\mathbf{r}_G$ and $\mathbf{Q}_G$ denote the global parameters and $\|\cdot\|_{\mathcal{F}}$ denotes the Frobenius norm. For $\mathbf{r}_G$, the relative difference of the mean square error along the independent dimensions are computed.

$$E_{rel}(\mathbf{Q}) = \frac{\|\mathbf{Q} - \mathbf{Q}_G\|_{\mathcal{F}}}{\|\mathbf{Q}_G\|_{\mathcal{F}}} \tag{20}$$

$$E_{rel}(\mathbf{r}) = \frac{\frac{1}{d}\sum_{j=1}^{d} \|\mathbf{r}_j - \mathbf{r}_{Gj}\|_2}{\frac{1}{d}\sum_{j=1}^{d} \|\mathbf{r}_{Gj}\|_2} \tag{21}$$

C. Results and Analysis

1) Performance: Firstly, changes in the hyper-priors do not yield any significant changes to the AUC score on the UNSW-NB15 dataset. Comparing the different anomaly scoring methodologies on EP-BRVFL-AE(C) in Table VI, the predictive variance score is the most consistent as the number of sites increase. RE can be used for further inspection of $f(\hat{\mathbf{x}})$ against $\hat{\mathbf{x}}$ to identify the type of anomaly by breaking it down to individual attributes. Also, it can be seen that $\mathcal{H}$ is dominated by RE. Table VI also shows that our method works well with hundreds of sites.

	UNSW-NB15			Shuttle		
No. Sites:	**500**	**100**	**1**	**500**	**100**	**1**
RE	89.88	89.91	89.93	86.67	94.19	99.74
$\phi(\hat{\mathbf{x}})^2$	89.99	89.99	89.98	95.30	96.64	99.39
$\mathcal{H}$	89.88	89.91	89.93	86.67	94.19	99.74

Table VI: AUC of different scoring methodologies on EP-BRVFL-AE(C) for different number of sites on UNSW-NB15 and Shuttle dataset.

Data at each site:	**UNSW-NB15**		**Shuttle**		**Aus Credit**	
	5	**10**	**5**	**10**	**5**	**10**
$\phi(\hat{\mathbf{x}})^2$	72.45	75.88	94.72	95.09	76.04	87.84
$\mathcal{H}$	70.60	74.04	95.54	95.22	77.70	87.65

Table VII: AUC of EP-BRVFL-AE(C) with small number of data points at each of the 10 sites.

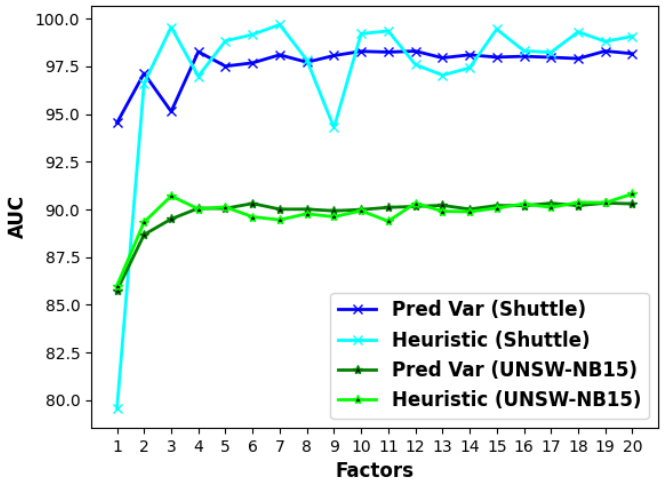


Figure 5: AUC against factor ζ for EP-BRVFL-AE(C) on the Shuttle and UNSW-NB15 datasets.

Fig 5 shows AUC against different multiplicative factors ζ on the nodes in the hidden layer. In general, the more nodes in the middle layer, the better results but it comes at a higher computational cost with a larger B. The results show that a factor of $\zeta = 5$ is sufficient.

Results of experimenting with a small number of data points in each site are shown in Table VII. The data points are sampled at random. Though the EP-BRVFL-AE(C) still performs with few data points, in this case, it is more beneficial to send the data to a central site as $B > n_k$ unless there are privacy concerns.

The EP-BRVFL-AE(C) is compared with various other approaches, namely Principal Component Analysis (PCA), K-Nearest Neighbours (KNN), Local Outlier Factor (LOF), Gaussian Mixture Model (GMM), Bayesian GMM, One Class Support Vector Machine (OCSVM), RVFL-AE and AE. Each of these models are trained locally using data present in each site as done in [12], [14]. The results over 10 and 50 sites are reported in Table VIII. The same test set is used across all sites and the average AUC and its standard deviation is reported. In the centralised approach, all data are sent to the central site where the model is trained.

Parameters for these other approaches are determined by commonly used rules of thumb. For PCA, the number of principal components (and for the AE the number of nodes in the hidden layer) is $\sqrt{d}+1$ [20]. The number of nearest-neighbours for KNN and LOF is $\sqrt{n_k}$. The GMM is trained using expectation maximisation [49]. The Bayesian GMM is trained using equation VI and Dirichlet process weight concentration [57]. The BRVFL-AE is trained locally and parameters are not shared using EP. Weights for the RVFL-AE are determined using (4). For BRVFL-AE and EP-BRVFL-AE, if the heuristic measure $\mathcal{H}$ performs better, it is reported in brackets; otherwise, the predictive variance measure is reported. These results are shown in Table VIII.

From Table VIII, the most consistent performing model over any number of sites is the EP-BRVFL-AE. The difference between the EP-BRVFL-AE and BRVFL-AE at the central site is due to the hyper-parameter optimisation. There is no standard deviation on EP-BRVFL-AE because the global parameters are shared across all sites. Sharing parameters using EP also improves the AUC result on the Australian Credit and Abalone datasets. Ranking the performance, the top three performing methods are EP-BRVFL-AE (C), BRVFL-AE and Bayesian GMM, which suggest that the Bayesian approaches are the best way to go about distributed training. As for the other methods, there is no clear consistent model. In some cases the standard deviation increases with number of sites, which shows that models trained only on local data can have different results. The Bayesian GMM and GMM show good overall performance but fail when there is not enough data at the local site, as can be seen with the Australian Credit dataset. Furthermore, the number of sites do not affect our method as the AUC score between 50 and 10 sites remain close.

To compare to some other distributed methods such as MVE-PCA [14], do-OCSVM, sparse doOCSVM [12], we perform EP on the same data sets used in those studies. From Table IX, the EP-BRVFL-AE(C) performs comparably to other methods in the literature. From Table IV, our method has lower complexities.

2) Fully Distributed: We mentioned previously that the EP-BRVFL-AE(C) gives the same result as EP-BRVFL-AE(D) when $\kappa = 1$. Table X gives the results with different number of sites and κ values. The network configurations are implemented at random for the various κ values and shown in Fig 6. For each network, the method is run for its Max Hops+2 iterations. The small standard deviation in AUC scores suggest that global solution at each site is almost similar to having a central site gathering all updates. This is further affirmed by the small average relative difference values, $E_{rel}(\mathbf{r})$ and

Datasets	UNSW-NB15			NSLKDD			Australian Credit Approval		
	Local		1	Local		1	Local		1
No. Sites:	50	10		50	10		50	10	
EP-BRVFL-AE(C)	**89.98**	**89.98**	**89.98**	**96.09**	96.09	96.09	**85.57**	**83.67**	80.82
BRVFL-AE	88.91 ± 1.0	89.81 ± 0.3	89.98	96.05 ± 0.2	96.16 ± 0.1	96.09	69.3(72.5)±10.4	73.44 ± 7.2	79.86
PCA	74.09 ± 1.7	73.75 ± 0.6	73.78	95.52 ± 0.3	95.50 ± 0.1	95.51	72.80 ± 10.9	71.31 ± 5.5	71.43
KNN	81.45 ± 0.8	83.74 ± 0.3	86.12	95.26 ± 0.2	94.92 ± 0.2	94.33	75.15 ± 9.2	81.13 ± 4.7	85.33
LOF	66.16 ± 3.4	81.32 ± 0.8	88.86	85.20 ± 2.8	83.91 ± 0.8	87.30	71.98 ± 12.1	79.05 ± 5.4	81.82
Bayesian GMM	88.77 ± 2.6	88.33 ± 2.5	87.29	95.59 ± 0.4	95.47 ± 0.4	95.25	50.0 ± 0.0	68.30 ± 5.7	82.08
GMM	83.70 ± 2.7	82.31 ± 0.8	84.62	95.45 ± 0.9	**96.17** ± 0.9	**96.59**	49.99 ± 0.1	50.00 ± 0.2	78.02
OCSVM	79.84 ± 1.1	79.94 ± 0.4	79.96	93.60 ± 0.2	93.65 ± 0.1	93.65	77.95 ± 8.9	83.51 ± 3.7	**86.84**
RVFL-AE	87.86 ± 0.9	88.73 ± 0.3	89.81	94.98 ± 0.2	95.21 ± 0.1	95.75	74.85 ± 9.8	71.22 ± 5.6	86.80
AE	81.19 ± 0.2	81.21 ± 0.1	81.21	91.69 ± 0.1	91.68 ± 0.1	91.68	77.33 ± 9.2	83.32 ± 2.6	85.07
Datasets	**Shuttle**			**Abalone**			**PageBlocks**		
	Local		1	Local		1	Local		1
No. Sites:	50	10		50	10		50	10	
EP-BRVFL-AE(C)	97.17	98.28	99.39	**75.70**	75.09	74.50	**97.34**	**97.44**	**97.56**
BRVFL-AE	95.39 ± 0.5	95.75 ± 0.5	97.26	72.74 ± 5.0	74.56 ± 1.9	74.62	96.35 ± 0.6	97.05 ± 0.3	97.49
PCA	84.95 ± 3.6	83.91 ± 3.4	83.14	68.12 ± 6.4	66.94 ± 2.8	65.73	95.04 ± 1.2	95.24 ± 0.6	95.34
KNN	97.92 ± 0.6	97.34 ± 0.6	98.05	59.32 ± 5.6	71.11 ± 3.0	77.98	95.04 ± 0.9	95.78 ± 0.3	96.36
LOF	98.34 ± 0.7	93.38 ± 2.3	98.79	51.87 ± 6.6	66.64 ± 2.9	74.15	96.22 ± 1.7	93.92 ± 1.0	95.06
Bayesian GMM	99.63 ± 0.1	99.23 ± 1.2	99.70	74.30 ± 4.5	**78.67** ± 1.1	**81.84**	96.04 ± 0.8	95.94 ± 0.5	96.49
GMM	**99.76** ± 0.1	**99.86** ± 0.0	**99.90**	60.10 ± 8.3	72.06 ± 2.7	81.38	92.89 ± 2.5	93.58 ± 2.1	94.19
OCSVM	96.51 ± 0.6	96.50 ± 0.2	96.49	55.02 ± 4.3	55.52 ± 2.0	55.78	96.25 ± 0.7	96.51 ± 0.2	96.59
RVFL-AE	94.33 ± 0.3	95.23 ± 0.2	95.01	64.52 ± 4.5	76.35 ± 1.8	76.69	97.02 ± 0.4	97.18 ± 0.2	95.38
AE	91.53 ± 0.2	91.53 ± 0.1	91.53	37.54 ± 3.5	36.37 ± 1.1	35.88	89.72 ± 1.5	89.62 ± 0.7	89.54

Table VIII: AUC over various datasets and methods. AUC using $\mathcal{H}$ is given in brackets if it performs better than $\phi(\hat{\mathbf{x}})^2$ for bayesian implementation. The data are randomly distributed. Both mean and standard deviation (mean ± standard deviation) are reported for methods where the model is learnt using local data at each site and results are averaged.

Table IX: Comparisons of AUC of different models

Datasets	Central		20 sites	
	MVE -PCA	EP-BRVFL -AE(C)	MVE -PCA	EP-BRVFL -AE(C)
Abalone	**83.28**	74.5	**82.73**	75.35
Shuttle	98.41	**99.66**	94.68	**97.83**
	Central		10 sites	
Aus Credit	80.77	**80.82**	73.98	**83.67**

Datasets	50 sites		
	do OCSVM	Sparse doOCSVM	EP-BRVFL -AE(C)
Abalone	63.41	64.52	**75.70**
PageBlocks	94.71	95.28	**97.38**

$E_{rel}(\mathbf{Q})$. The results show that the method works irrespective of κ. Similar results are observed on the other datasets.

3) Biased Partitions of Data: In some networks, distribution of data from individual sites may be different. Furthermore, the number of data points could vary widely. We experiment on the worst-case scenario where data in each site have different profiles. Using a GMM, we split the data into 10 such that each site contains data from a separate Gaussian component. Other studies on distributed training have not evaluated their methods in this manner. We use the UNSW-NB15 dataset to report the results in Table XI.

Fig 7a shows mean AUC increasing and Fig 7b shows standard deviation amongst AUC over all sites decreasing as updates are being received at each iteration. Figs. 7c and 7d show relative errors decreasing with respect to EP-BRVFL-AE(C). The results show that at the iteration after the Max Hops of the network, the solutions converge. This also further validates that the first update from each site is the most important. Hence, s in the computational complexity is at most the Max Hops value. From Table XI and Fig 7, the solutions for the worst case scenario on the poorest distributed network for 10 sites of $\kappa = 0.222$ still converge. This confirms that the solution will converge for any network configurations.

Furthermore, in our test, EP-BRVFL-AE(C) achieves an AUC of 89.76 and 89.46 for $\phi(\hat{\mathbf{x}})^2$ and $\mathcal{H}$ measures respectively with GMM split. An important observation is that EP-BRVFL-AE(D) with $\kappa = 1$ achieves the same result in Table XI, despite the biased and uneven partition of data. This implies that updates from each site can still be combined using EP to build the model. Hence, logically, the model is robust under transmission delays. The computation at each site can continue and the update can be included in the following iteration. We simulate the worst-case scenario where updates from each site arrive one at a time and the computation for the global parameters is performed after each arrival. Fig 8a shows how the global parameters converge to the scenario where all updates arrive together at the central site. The AUC score increases as each update from the site is included as depicted in Fig 8b. Hence, our method allows asynchronous updates to converge with the varied information received at each site regardless of network configurations. This is in fact possible because the model is simple and theoretically robust with closed form solutions.

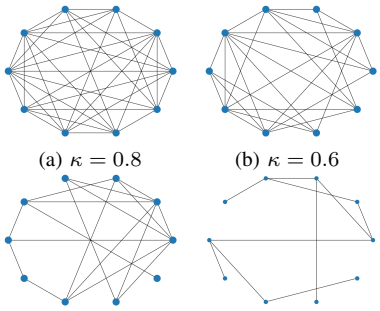
The value of δ remains close to the initial value of 1 when the data are evenly distributed. For the case of GMM split, Fig 9 shows the minimum and average δ values for $\mathbf{Q}_{-k}$ to remain positive definite at different κ values. Most of the time, as δ remains unchanged, $t = 1$ in the computational complexity.

D. Comparison with Multi-Layer Neural Network, Federated Learning and Datasets with Fewer Anomalies

EP-BRVFL-AE gives a good model for distributed training and anomaly detection. The SLFN provides a fast, efficient and simple solution to work at the edge of the network. One of the essentials it lacks is depth in the neural network.

Network				Convergence		AUC (Mean ± Std. Dev.)	
Sites	κ	ADPS	MHs	$E_{rel}(\mathbf{r})$	$E_{rel}(\mathbf{Q})$	$\phi(\hat{\mathbf{x}})^2$	$\mathcal{H}$
10	0.8	7.2	2	0.078	0.003	89.98±0.00	89.93±0.00
10	0.6	5.4	3	0.045	0.002	89.99±0.00	89.93±0.00
10	0.4	3.6	3	0.061	0.002	89.99±0.00	89.93±0.00
10	0.222	1.11	6	0.018	0.001	89.99±0.00	89.93±0.00
50	0.8	39.2	2	0.169	0.006	89.98±0.00	89.92±0.00
50	0.4	19.6	3	0.123	0.005	89.98±0.00	89.92±0.00
50	0.2	9.8	4	0.088	0.003	89.98±0.00	89.98±0.00

Table X: Performance of EP-BRVFL-AE(D) on UNSW-NB15 dataset. Average values over the sites on the last iteration are reported for $E_{rel}(\mathbf{r})$ and $E_{rel}(\mathbf{Q})$ against the solution for EP-BRVFL-AE(C). Mean and standard deviation of Area under ROC curves (AUC) using both $\phi(\hat{\mathbf{x}})$ and $\mathcal{H}$ over the distributed sites are reported.

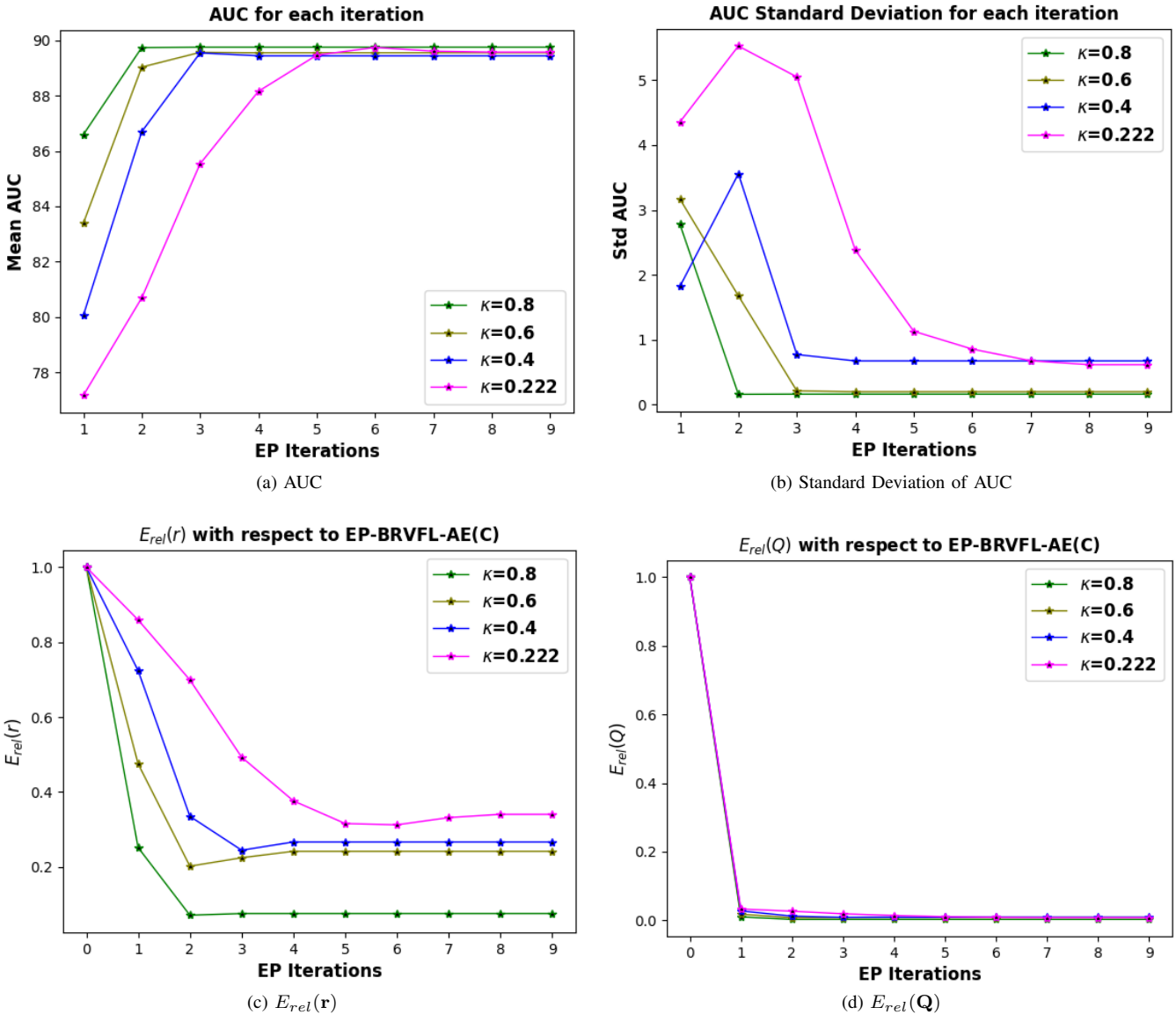


(a) $\kappa = 0.8$ (b) $\kappa = 0.6$ (c) $\kappa = 0.4$ (d) $\kappa = 0.222$

Figure 6: Networks with 10 sites

Network				Convergence		AUC (Mean ± Standard Deviation)	
Sites	κ	ADPS	Max Hops	$E_{rel}(\mathbf{r})$	$E_{rel}(\mathbf{Q})$	$\phi(\hat{\mathbf{x}})^2$	$\mathcal{H}$
10	1	9	1	0.0023	0.000	89.76 ± 0.000	89.46 ± 0.000
10	0.8	7.2	2	0.0758	0.0029	89.75 ± 0.162	89.40 ± 0.141
10	0.6	5.4	3	0.2419	0.0084	89.55 ± 0.200	89.34 ± 0.137
10	0.4	3.6	3	0.2668	0.0091	89.44 ± 0.674	89.19 ± 0.604
10	0.222	1.11	6	0.3406	0.0080	89.57 ± 0.616	89.78 ± 0.683

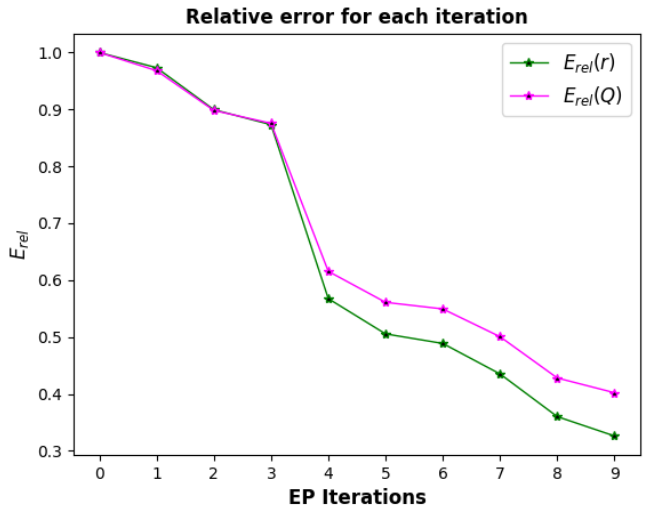
Table XI: Performance of EP-BRVFL-AE(D) on UNSW-NB15 with biased and uneven partitions of data.



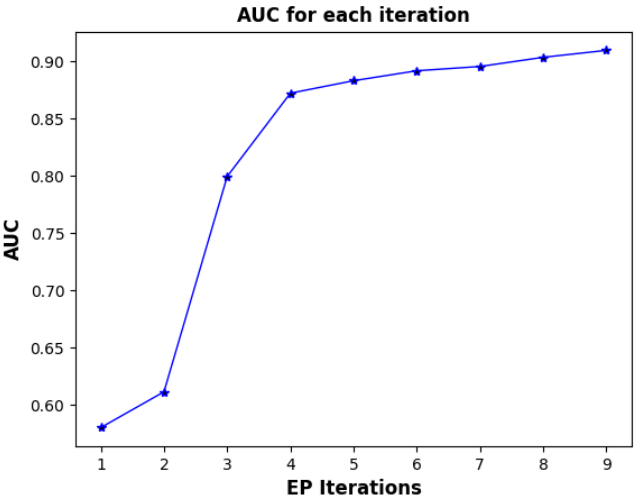
(a) AUC (b) Standard Deviation of AUC

(c) $E_{rel}(\mathbf{r})$ (d) $E_{rel}(\mathbf{Q})$

Figure 7: EP-BRVFL-AE(D) with 10 sites evaluated at each EP iteration on the UNSW-NB15 dataset with GMM split.

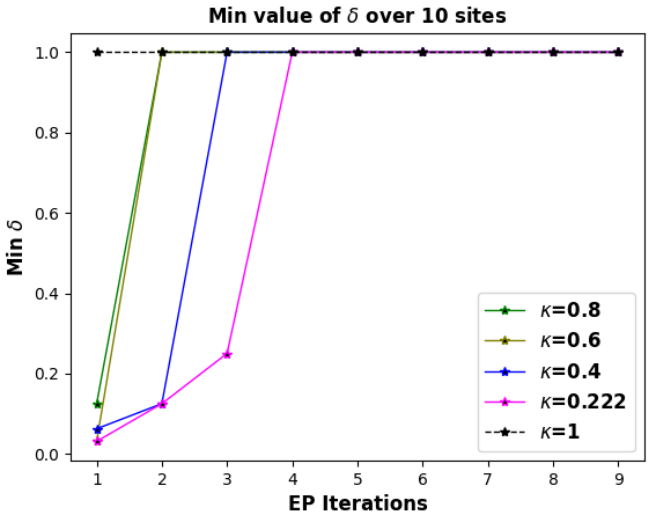


(a) $E_{rel}(\mathbf{r})$ and $E_{rel}(\mathbf{Q})$

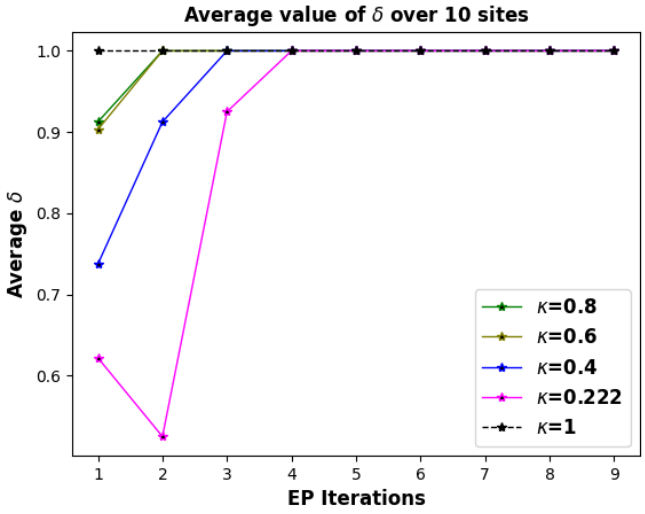


(b) AUC

Figure 8: EP-BRVFL-AE(C) with 10 sites where updates are combined one by one on UNSW-NB15 with GMM split.



(a) Minimum δ



(b) Average δ

Figure 9: (a) Minimum and (b) Average of δ over 10 sites during each EP iteration

One immediate implication is that with more hidden layers in the neural network with non-linear activations, Bayesian inference is no longer analytically tractable. Hence a closed form solution does not exist. In algorithms 1 and 2, the computation of the tilted distribution needs to be replaced by approximation methods such as Variational Inference (VI) [49] and a Bayesian AutoEncoder (BAE) is compared with BRVFL-AE. An AE with 3 hidden layers is used. A similar AE is trained using federated learning [34] for comparison.

The AUC scores of EP-BRVFL-AE(C), EP-BAE(C) and FL-AE (Federated Learning AutoEncoder [34]) with data at 5 sites are shown in Table XII. ADVI [58] was used to compute the tilted distribution for the BAE. The MAP estimate of the posterior is used as a one value estimate of the weights and the RE is used as the anomaly measure. The global solution is used for FL-AE. The results show that a closed form solution with SLFN performs better than both EP-BAE and FL-AE for distributed training and that the tractability of computations contributes to the performance especially in distributed training. Multi-layer neural networks also require immense computational complexity and power requirements. The training times of EP-BAE, FL-AE and EP-BRVFL-AE was in order of hours, minutes and seconds respectively. FL-AE does not have a Bayesian component in comparison to the others and it took multiple iterations before convergence was achieved. In comparison, EP-BRVFL-AE converged in 1 iteration as it is in closed form. The aim of this work is to support all types of edge devices, both smartphones with GPUs and Machine Learning (ML) accelerators, as well as the increasing number of devices that are not enhanced with ML accelerators such as IoT devices and wireless sensors. Hence, BRVFL-AE is certainly more suitable for edge devices without ML accelerators for real-time applications.

In anomaly detection, the outlier or anomaly points may be rare, typically around 1-5% of the overall dataset. Hence, we evaluate our method with randomly selected anomaly

Model	AUC	Training Time
EP-BAE (C) with 5 sites	76.54	16.11 hours
FL-AE (C) with 5 sites	85.98	3.32 minutes
EP-BRVFL-AE (C) with 5 sites	**89.93**	**10.76 seconds**

Table XII: AUC scores and training times on UNSW-NB15 dataset

Datasets	AUC	Average Precision
UNSW-NB15	99.87	94.40
NSLKDD	99.98	99.01
Australian Credit	98.28	71.55
Shuttle	98.81	91.09
Abalone	96.27	59.52
Page Blocks	99.87	96.68
Fraud	97.28	87.30
Campaign	99.80	83.72

Table XIII: AUC and Average Precision scores of EP-BRVFL-AE(C) with 5 sites on datasets with anomalous points being only 3% of the overall dataset.

points forming 3% of the overall dataset for testing. We also evaluate on two other suitable datasets namely Fraud [59] and Campaign [60]. As anomaly points are rare, the average precision measure is used to evaluate the results. Average precision denotes the area under the Precision-Recall curve. The results are shown in Table XIII. Table XIV compares the other ensemble methods with the two datasets. Bayesian GMM and GMM perform well, though with variability, when there are few anomalies but not as well compared to the results in Table VIII. EP-BRVFL-AE(C) still performs well and consistently in many situations.

VI. Conclusion and Future Work

In this work, we have described a novel EP-BRVFL-AE for anomaly detection in Edge AI networks. The model is trained in a distributed manner without having to share raw data from each site. Only changes in the posterior parameters of the EP-BRVFL-AE weights are shared. Furthermore, the use of SLFN and the conjugate prior ensues a closed-form solution which achieves rapid convergence. The longest wait time is the number of Max Hops within a network. Moreover, asynchronous update is also possible.

We evaluated the EP-BRVFL-AE against other methods in the literature and in the distributed setting with varying network densities. The detection performance is marginally better, the computational complexity is greatly reduced and we have added measures of variances to understand the data distribution better. We performed worst case scenario analysis by splitting the data in a biased and uneven manner where other methods in the literature do not. The algorithm performs well under all of the mentioned scenarios and it is more suitable in edge devices with resource constraints and for applications which require quick decision making.

The key findings of this paper are summarised as follows:

- A SLFN reduces computational complexity, allows for quick training, inference and feasibility at the edge.
- A BRVFL enables closed-form solutions for tractability and quick aggregation of model parameters.
- The Bayesian approach accounts for variance in the data at distributed sites. EP incorporates it during aggregation and allows a robust model to be built with biased partitions of data and in an asynchronous manner.
- EP allows a robust model to be built under inhomogeneous network densities and without a central site.

	Fraud		Campaign	
Methods	**AUC**	**Avr Prec**	**AUC**	**Avr Prec**
EP-BRVFL-AE(C)	97.28	87.30	99.80	83.72
BRVFL-AE	97.36±0.0	87.64±0.4	61.92±3.3	3.72±0.3
PCA	96.34±0.1	82.57±0.1	47.15±1.7	2.58±0.1
KNN	95.77±0.0	82.45±0.0	50.89±0.2	2.93±0.0
LOF	95.76±0.1	83.91±0.2	64.14±1.4	3.83±0.2
Bayesian GMM	97.66±0.1	85.01±3.5	99.76±0.0	98.91±0.0
GMM	97.54±0.2	86.16±3.6	99.81±0.1	96.84±6.3
OCSVM	95.78±0.0	80.19±0.1	50.58±0.1	2.94±0.0
RVFL-AE	97.03±0.1	79.62±4.7	64.80±0.7	4.06±0.1
AE	93.68±0.0	71.50±0.1	48.36±0.2	2.79±0.0

Table XIV: AUC and Average Precision (Avr Prec) scores of various methods on the Fraud and Campaign datasets on 5 sites with anomalous points being only 3% of the overall dataset.

As future work, different prior assumptions can be explored. In this paper, we have used one value for σ^2 and one value for γ obtained through EP. This can be further extended to using σ_j^2 for each dimension j and γ_l for each weight. The former will lead to a different covariance matrix Σ_j for each dimension and the latter is also known as Automatic Relevance Determination [22], [49]. Instead of assuming a Gaussian distribution for the likelihood and prior in (6) and (7), other distributions such as the student's t-distribution can be used.

Anomaly detection encompasses a wide range of applications. Distributed anomaly detection models could be used to detect malicious router updates in networking or to detect new objects in self-driving vehicles. Hence, it involves smart engineering to design the model for specific applications. In this aspect, the method can also be further studied with communication protocols in resource constrained environments to manage events and trust between devices [61] and time taken to train and detect anomalies can be compared with various edge devices' power, OS and RAM.

References

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[2] G. A. Kaissis, M. R. Makowski, D. Rückert, and R. F. Braren, "Secure, privacy-preserving and federated machine learning in medical imaging," *Nature Machine Intelligence*, pp. 1–7, 2020.

[3] L. Wang and B. K. Horn, "Time-to-Contact control for safety and reliability of self-driving cars," in *Proceedings of the IEEE International Smart Cities Conference*, Wuxi, China, 14-17 Sept 2017.

[4] H. El-Sayed, S. Sankar, M. Prasad, D. Puthal, A. Gupta, M. Mohanty, and C.-T. Lin, "Edge of things: the big picture on the integration of edge, IoT and the cloud in a distributed computing environment," *IEEE Access*, vol. 6, pp. 1706–1717, 2017.

[5] W. Sultani, C. Chen, and M. Shah, "Real-world anomaly detection in surveillance videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, 18-23 June 2018, pp. 6479–6488.

[6] M. Xie, J. Hu, and S. Guo, "Segment-based anomaly detection with approximated sample covariance matrix in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 2, pp. 574–583, 2014.

[7] L. Lyu, J. C. Bezdek, X. He, and J. Jin, "Fog-embedded deep learning for the internet of things," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4206–4215, 2019.

[8] L. Lyu, J. Jin, S. Rajasegarar, X. He, and M. Palaniswami, "Fog-empowered anomaly detection in IoT using hyperellipsoidal clustering," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1174–1184, 2017.

[9] N. Peri, P. Khorramshahi, S. S. Rambhatla, V. Shenoy, S. Rawat, J.-C. Chen, and R. Chellappa, "Towards real-time systems for vehicle re-identification, multi-camera tracking, and anomaly detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 622–623.

[10] Q. Li, Y. Liu, Z. Liu, P. Zhang, and C. Pang, "Efficient Forwarding Anomaly Detection in Software-Defined Networks," *IEEE Transactions on Parallel and Distributed Systems*, 2021.

[11] S. Zhang, R. Benenson, M. Omran, J. Hosang, and B. Schiele, "Towards reaching human performance in pedestrian detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 973–986, 2017.

[12] X. Miao, Y. Liu, H. Zhao, and C. Li, "Distributed online one-class support vector machine for anomaly detection over networks," *IEEE Transactions on Cybernetics*, vol. 49, no. 4, pp. 1475–1488, 2019.

[13] Y.-L. Tsou, H.-M. Chu, C. Li, and S.-W. Yang, "Robust Distributed Anomaly Detection Using Optimal Weighted One-Class Random Forests," in *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, Singapore, 17-20 Nov 2018, pp. 1272–1277.

[14] C. O'Reilly, A. Gluhak, and M. A. Imran, "Distributed anomaly detection using minimum volume elliptical principal component analysis," *IEEE Trans on Knowledge and Data Engineering*, vol. 28, no. 9, 2016.

[15] W. Neiswanger, C. Wang, and E. P. Xing, "Asymptotically exact, embarrassingly parallel MCMC," in *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI)*, Quebec, Canada, 23-27 July 2014, pp. 623–632.

[16] B. Safarinejadian and M. E. Estahbanati, "Consensus filter-based distributed variational Bayesian algorithm for flow and speed density prediction with distributed traffic sensors," *IEEE Systems Journal*, vol. 11, no. 4, pp. 2939–2948, 2015.

[17] A. Gelman, A. Vehtari, P. Jylänki, T. Sivula, D. Tran, S. Sahai, P. Blomstedt, J. P. Cunningham, D. Schiminovich, and C. Robert, "Expectation propagation as a way of life: A framework for Bayesian inference on partitioned data," *arXiv preprint arXiv:1412.4869*, 2017.

[18] K. S. Chahal, M. S. Grover, K. Dey, and R. R. Shah, "A hitchhiker's guide on distributed training of deep neural networks," *Journal of Parallel and Distributed Computing*, vol. 137, pp. 65–76, 2020.

[19] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *Proceedings of the IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, Atlanta, GA, USA, 5-8 June 2017, pp. 328–339.

[20] V. L. Cao, M. Nicolau, and J. McDermott, "Learning Neural Representations for Network Anomaly Detection," *IEEE Transactions on Cybernetics*, vol. 49, no. 8, pp. 3074–3087, Aug 2019.

[21] Z. Chen, C. K. Yeo, B. S. Lee, and C. T. Lau, "Autoencoder-based network anomaly detection," in *Proceedings of the Wireless Telecommunications Symposium (WTS)*, Phoenix, AZ, USA, 17-20 Apr 2018.

[22] S. Scardapane, D. Wang, and A. Uncini, "Bayesian random vector functional-link networks for robust data modeling," *IEEE Transactions on Cybernetics*, vol. 48, no. 7, pp. 2049–2059, 2017.

[23] Y. Ren, X. Qiu, P. N. Suganthan, and G. Amaratunga, "Detecting Wind Power Ramp with Random Vector Functional Link (RVFL) Network," in *Proceedings of the IEEE Symposium Series on Computational Intelligence*, Cape Town, South Africa, 7-10 Dec 2015, pp. 687–694.

[24] L. Tang, Y. Wu, and L. Yu, "A non-iterative decomposition-ensemble learning paradigm using RVFL network for crude oil price forecasting," *Applied Soft Computing*, vol. 70, pp. 1097–1108, 2018.

[25] M. Pratama, P. P. Angelov, E. Lughofer, and M. J. Er, "Parsimonious random vector functional link network for data streams," *Information Sciences*, vol. 430, pp. 519–537, 2018.

[26] S. Scardapane, D. Wang, M. Panella, and A. Uncini, "Distributed learning for random vector functional-link networks," *Information Sciences*, vol. 301, pp. 271–284, 2015.

[27] K. Shen, Z. Jing, and P. Dong, "A consensus nonlinear filter with measurement uncertainty in distributed sensor networks," *IEEE Signal Processing Letters*, vol. 24, no. 11, pp. 1631–1635, 2017.

[28] B. Liu, Z. Ding, and C. Lv, "Distributed Training for Multi-Layer Neural Networks by Consensus," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 5, pp. 1771–1778, 2020.

[29] M. Langer, Z. He, W. Rahayu, and Y. Xue, "Distributed training of deep learning models: A taxonomic perspective," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 12, pp. 2802–2818, 2020.

[30] C. Hardy, E. Le Merrer, and B. Sericola, "MD-GAN: Multi-discriminator generative adversarial networks for distributed datasets," in *Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, Rio de Janeiro, Brazil, Brazil, 20-24 May 2019.

[31] H. Wang, D. Niu, and B. Li, "Distributed machine learning with a serverless architecture," in *Proceedings of the IEEE INFOCOM*, Paris, France, 29 April-2 May 2019, pp. 1288–1296.

[32] S. Shi, X. Chu, and B. Li, "MG-WFBP: Efficient data communication for distributed synchronous SGD algorithms," in *Proceedings of the IEEE INFOCOM*, Paris, France, 29 April-2 May 2019.

[33] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *Proceedings of the IEEE INFOCOM*, Honolulu, HI, USA, 16-19 Apr 2018.

[34] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*, 2017, pp. 1273–1282.

[35] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[36] N. Kourtellis, K. Katevas, and D. Perino, "FLaaS: Federated Learning as a Service," in *Proceedings of the 1st Workshop on Distributed Machine Learning*, 2020, pp. 7–13.

[37] D. Liu, "Accelerating Intra-Party Communication in Vertical Federated Learning with RDMA," in *Proceedings of the 1st Workshop on Distributed Machine Learning*, 2020, pp. 14–20.

[38] M. Duan, D. Liu, X. Chen, R. Liu, Y. Tan, and L. Liang, "Self-balancing federated learning with global imbalanced data in mobile systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 59–71, 2020.

[39] Q. Jia, L. Guo, Z. Jin, and Y. Fang, "Preserving model privacy for machine learning in distributed systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 8, pp. 1808–1822, 2018.

[40] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A Survey on Federated Learning for Resource-Constrained IoT Devices," *IEEE Internet of Things Journal*, 2021.

[41] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, "A review of applications in federated learning," *Computers & Industrial Engineering*, p. 106854, 2020.

[42] Y. Yan, L. Cao, and E. A. Rundensteiner, "Distributed Top-N local outlier detection in big data," in *Proceedings of the IEEE International Conference on Big Data (Big Data)*, Boston, MA, USA, 11-14 Dec 2017, pp. 827–836.

[43] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "DÏoT: A federated self-learning anomaly detection system for IoT," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 756–767.

[44] Y. Liu, S. Garg, J. Nie, Y. Zhang, Z. Xiong, J. Kang, and M. S. Hossain, "Deep anomaly detection for time-series data in industrial iot: A communication-efficient on-device federated learning approach," *IEEE Internet of Things Journal*, 2020.

[45] Y. Zhao, J. Chen, D. Wu, J. Teng, and S. Yu, "Multi-task network anomaly detection using federated learning," in *Proceedings of the tenth international symposium on information and communication technology*, 2019, pp. 273–279.

[46] L. Hasenclever, S. Webb, T. Lienart, S. Vollmer, B. Lakshminarayanan, C. Blundell, and Y. W. Teh, "Distributed Bayesian learning with stochastic natural gradient expectation propagation and the posterior server," *The Jrnl of Machine Learning Research*, vol. 18, pp. 3744–3780, 2017.

[47] M. Xu, B. Lakshminarayanan, Y. W. Teh, J. Zhu, and B. Zhang, "Distributed Bayesian posterior sampling via moment sharing," in *Advances in Neural Information Processing Systems*, 2014, pp. 3356–3364.

[48] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, *Bayesian data analysis*. Chapman and Hall/CRC, 2013.

[49] C. M. Bishop, "Pattern recognition and machine learning springer-verlag new york," *Inc. Secaucus, NJ, USA*, 2006.

[50] M. Opper and O. Winther, "Gaussian processes for classification: Mean-field algorithms," *Neural computation*, vol. 12, pp. 2655–2684, 2000.

[51] T. P. Minka, "Expectation propagation for approximate Bayesian inference," *Proc. of 17th Conf. in Uncertainty in Artificial Intelligence*, 2001.

[52] M. Seeger, "Expectation propagation for exponential families," UC Berkeley, Tech. Rep., 2005.

[53] N. Moustafa and J. Slay, "The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," *Information Security Journal: A Global Perspective*, vol. 25, no. 1-3, pp. 18–31, 2016.

[54] L. Dhanabal and S. Shantharajah, "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms," *Int'l. Jrnl.*

of Advanced Research in Computer and Communication Engineering, vol. 4, no. 6, pp. 446–452, 2015.

[55] D. Dua and C. Graff, "UCI Machine Learning Repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[56] M.-Y. Su, "Real-time anomaly detection systems for Denial-of-Service attacks by weighted k-nearest-neighbor classifiers," *Expert Systems with Applications*, vol. 38, no. 4, pp. 3492–3498, 2011.

[57] D. M. Blei, M. I. Jordan *et al.*, "Variational inference for Dirichlet process mixtures," *Bayesian analysis*, vol. 1, no. 1, pp. 121–143, 2006.

[58] A. Kucukelbir, D. Tran, R. Ranganath, A. Gelman, and D. M. Blei, "Automatic differentiation variational inference," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 430–474, 2017.

[59] A. Dal Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi, "Calibrating probability with undersampling for unbalanced classification," in *2015 IEEE Symposium Series on Computational Intelligence*. IEEE, 2015, pp. 159–166.

[60] S. Moro, P. Cortez, and P. Rita, "A data-driven approach to predict the success of bank telemarketing," *Decision Support Systems*, vol. 62, pp. 22–31, 2014.

[61] C. Esposito, A. Castiglione, F. Palmieri, M. Ficco, C. Dobre, G. V. Iordache, and F. Pop, "Event-based sensor data exchange and fusion in the Internet of Things environments," *Journal of Parallel and Distributed Computing*, vol. 118, pp. 328–343, 2018.