Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications

Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei*
Tsinghua University

Yang Feng, Jie Chen, Zhaogang Wang, Honglin Qiao
Alibaba Group

ABSTRACT

To ensure undisrupted business, large Internet companies need to closely monitor various KPIs (*e.g.*, Page Views, number of online users, and number of orders) of its Web applications, to accurately detect anomalies and trigger timely troubleshooting/mitigation. However, anomaly detection for these seasonal KPIs with various patterns and data quality has been a great challenge, especially without labels. In this paper, we proposed *Donut*, an unsupervised anomaly detection algorithm based on VAE. Thanks to a few of our key techniques, *Donut* greatly outperforms a state-of-arts supervised ensemble approach and a baseline VAE approach, and its best F-scores range from 0.75 to 0.9 for the studied KPIs from a top global Internet company. We come up with a novel KDE interpretation of reconstruction for *Donut*, making it the first VAE-based anomaly detection algorithm with solid theoretical explanation.

ACM Reference Format:
Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei, Yang Feng, Jie Chen, Zhaogang Wang and Honglin Qiao. 2018. Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications. In *WWW 2018: The 2018 Web Conference, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3178876.3185996

1 INTRODUCTION

To ensure undisrupted business, large Internet companies need to closely monitor various KPIs (key performance indicators) of its Web applications, to accurately detect anomalies and trigger timely troubleshooting/mitigation. KPIs are time series data, measuring metrics such as Page Views, number of online users, and number of orders. Among all KPIs, the most ones are business-related KPIs (the focus of this paper), which are heavily influenced by user behavior and schedule, thus roughly have seasonal patterns occurring at regular intervals (*e.g.*, daily and/or weekly). However, anomaly detection for these seasonal KPIs with various patterns and data quality has been a great challenge, especially without labels.

A rich body of literature exist on detecting KPI anomalies [1, 2, 5–8, 17–19, 21–25, 27, 29, 32, 33, 37, 38]. As discussed in § 2.2, existing anomaly detection algorithms suffer from the hassle of algorithm picking/parameter tuning, heavy reliance on labels, unsatisfying performance, and/or lack of theoretical foundations.

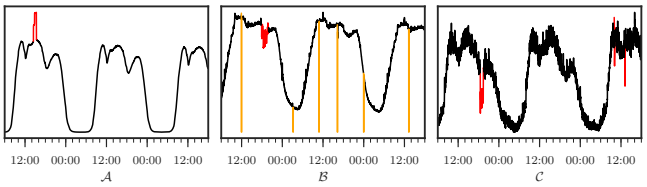


Figure 1: 2.5-day-long fragments of the seasonal KPI datasets in our paper, with anomalies in red color and missing points (filled with zeros) in orange. Within each dataset, there are variations for the same time slot in different days.

In this paper, we propose *Donut*, an unsupervised anomaly detection algorithm based on Variational Auto-Encoder (a representative deep generative model) with solid theoretical explanation, and this algorithm can work when there are no labels at all, and can take advantage of the occasional labels when available.

The contributions of this paper can be summarized as follows.

- The three techniques in *Donut*, Modified ELBO and Missing Data Injection for training, and MCMC Imputation for detection, enable it to greatly outperform state-of-art supervised and VAE-based anomaly detection algorithms. The best F-scores of unsupervised *Donut* range from 0.75 to 0.9 for the studied KPIs from a top global Internet company.
- For the first time in the literature, we discover that adopting VAE (or generative models in general) for anomaly detection requires training on both normal data *and abnormal data*, contrary to common intuition.
- We propose a novel KDE interpretation in z-space for *Donut*, making it the first VAE-based anomaly detection algorithm with solid theoretical explanation unlike [2, 33]. This interpretation may benefit the design of other deep generative models in anomaly detection. We discover a *time gradient effect* in latent z-space, which nicely explain *Donut*'s excellent performance for detecting anomalies in seasonal KPIs.

2 BACKGROUND AND PROBLEM

2.1 Context and Anomaly Detection in General

In this paper, we focus on business-related KPIs. These time series are heavily influenced by user behavior and schedule, thus roughly have *seasonal* patterns occurring at regular intervals (*e.g.*, daily and/or weekly). On the other hand, the *shapes* of the KPI curves at each repetitive cycle are *not* exactly the same, since user behavior can vary across days. We hereby name the KPIs we study **"seasonal**

*Dan Pei is the corresponding author.
[1] An implementation of *Donut* is published at https://github.com/korepwx/donut
[2] A longer version of the paper can be found at [36].

WWW 2018, April 23–27, 2018, Lyon, France

ACM ISBN 978-1-4503-5639-8/18/04.
https://doi.org/10.1145/3178876.3185996

KPIs with local variations". Examples of such KPIs are shown in Fig 1. Another type of local variation is the increasing trend over days, as can be identified by Holt-Winters [38] and Time Series Decomposition [6]. An anomaly detection algorithm may not work well unless these local variations are properly handled.

In addition to the seasonal patterns and local variations of the KPI *shapes*, there are also noises on these KPIs, which we assume to be independent, zero-mean Gaussian at every point. The exact values of the Gaussian noises are meaningless, thus we only focus on the statistics of these noises, *i.e.*, the variances of the noises.

We can now formalize the "normal patterns" of seasonal KPIs as a combination of two components: (1) the seasonal patterns with local variations, and (2) the statistics of the Gaussian noises.

We use "anomalies" to denote the recorded points which do not follow normal patterns (*e.g.*, sudden spikes and dips) , while using "abnormal" to denote both anomalies and missing points. See Fig 1 for examples of both anomalies and missing points. Because the KPIs are monitored periodically (*e.g.*, every minute), missing points are recorded as "null" (when the monitoring system does not receive the data) and thus are straightforward to identify. We thus focus on detecting anomalies for the KPIs.

Because operators need to deal with the anomalies for troubleshooting/mitigation, some of the anomalies are anecdotally labeled. Note that such occasional labels' coverage of anomalies are far from what's needed for typical supervised learning algorithms.

Anomaly detection on KPIs can be formulated as follows: for any time t, given historical observations $x_{t-T+1}, \ldots, x_t$, determine whether an anomaly occurs (denoted by $y_t = 1$). An anomaly detection algorithm typically computes a real-valued score indicating the certainty of having $y_t = 1$, *e.g.*, $p(y_t = 1 | x_{t-T+1}, \ldots, x_t)$, instead of directly computing y_t. Human operators can then affect whether to declare an anomaly by choosing a threshold, where a data point with a score exceeding this threshold indicates an anomaly.

2.2 Previous Work

Traditional statistical models. Over the years, quite a few anomaly detectors based on traditional statistical models (*e.g.*, [6, 17, 18, 22, 24, 25, 29, 37, 38], mostly time series models) have been proposed to compute anomaly scores. Because these algorithms typically have simple assumptions for applicable KPIs, expert's efforts need to be involved to pick a suitable detector for a given KPI, and then fine-tune the detector's parameters based on the training data. Simple ensemble of these detectors, such as majority vote [8] and normalization [32], do not help much either according to [23]. As a result, these detectors see only limited use in the practice.

Supervised ensemble approaches. To circumvent the hassle of algorithm/parameter tuning for traditional statistical anomaly detectors, supervised ensemble approaches, EGADS [19] and Opprentice [23], were proposed. They train anomaly classifiers using the user feedbacks as labels and using anomaly scores output by traditional detectors as features. Both EGADS and Opprentice showed promising results, but they heavily rely on good labels (much more than the anecdotal labels accumulated in our context), which is generally not feasible in large scale applications. Furthermore, running multiple traditional detectors to extract features during detection introduces lots of computational cost, which is a practical concern.

Unsupervised approaches and deep generative models. Recently, there is a rising trend of adopting unsupervised machine learning algorithms for anomaly detection, *e.g.*, one-class SVM [1, 7], clustering based methods [9] like K-Means [26] and GMM [21], KDE [27], and VAE [2] and VRNN [33]. The philosophy is to focus on normal patterns instead of anomalies: since the KPIs are typically composed mostly of normal data, models can be readily trained even without labels. Roughly speaking, they all first recognize "normal" regions in the original or some latent feature space, and then compute the anomaly score by measuring "how far" an observation is from the normal regions.

Along this direction, we are interested in deep generative models for the following reasons. First, learning normal patterns can be seen as learning the distribution of training data, which is a topic of generative models. Second, great advances have been achieved recently to train generative models with deep learning techniques, *e.g.*, GAN [13] and deep Bayesian network [4, 35]. The latter is family of deep generative models, which adopts the graphical [28] model framework and variational techniques [3], with the VAE [16, 30] as a representative work. Third, despite deep generative model's great promise in anomaly detection, existing VAE-based anomaly detection method [2] was not designed for KPIs (time series), and does not perform well in our settings (see § 4), and there is no theoretical foundation to back up its designs of deep generative models for anomaly detection (see § 5). Fourth, simply adopting the more complex models [33] based on VRNN shows long training time and poor performance in our experiments. Fifth, [2] assumes training only on clean data, which is infeasible in our context, while [33] does not discuss this problem.

2.3 Problem Statement

In summary, existing anomaly detection algorithms suffer from the hassle of algorithm picking/parameter tuning, heavy reliance on labels, unsatisfying performance, and/or lack of theoretical foundations. Existing approaches are either unsupervised, or supervised but depending heavily on labels. However, in our context, labels are occasionally available although far from complete, which should be somehow taken advantage of.

The problem statement of this paper is as follows. **We aim at an unsupervised anomaly detection algorithm based on deep generative models with solid theoretical explanation, and this algorithm can take advantage of the occasionally available labels.** Because VAE is a basic building block of deep Bayesian network, we chose to start our work with VAE.

2.4 Background of Variational Auto-Encoder

Deep Bayesian networks use neural networks to express the relationships between variables, such that they are no longer restricted to simple distribution families, thus can be easily applied to complicated data. Variational inference techniques [12] are often adopted in training and prediction, which are efficient methods to solve posteriors of the distributions derived by neural networks.

VAE is a deep Bayesian network. It models the relationship between two random variables, latent variable $\mathbf{z}$ and visible variable $\mathbf{x}$. A prior is chosen for $\mathbf{z}$, which is usually multivariate unit Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$. After that, $\mathbf{x}$ is sampled from $p_\theta(\mathbf{x}|\mathbf{z})$, which is derived from a neural network with parameter θ. The exact form

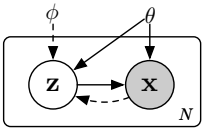


Figure 2: Architecture of VAE. The prior of z is regarded as part of the generative model (solid lines), thus the whole generative model is denoted as $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})$. The approximated posterior (dashed lines) is denoted as $q_\phi(\mathbf{z}|\mathbf{x})$.

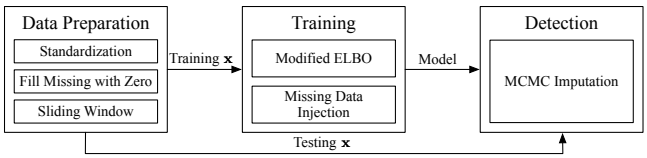


Figure 3: Overall architecture of *Donut*.

of $p_\theta(\mathbf{x}|\mathbf{z})$ is chosen according to the demand of task. The true posterior $p_\theta(\mathbf{z}|\mathbf{x})$ is intractable by analytic methods, but is necessary for training and often useful in prediction, thus the variational inference techniques are used to fit another neural network as the approximation posterior $q_\phi(\mathbf{z}|\mathbf{x})$. This posterior is usually assumed to be $\mathcal{N}(\boldsymbol{\mu}_\phi(\mathbf{x}), \boldsymbol{\sigma}^2_\phi(\mathbf{x}))$, where $\boldsymbol{\mu}_\phi(\mathbf{x})$ and $\boldsymbol{\sigma}_\phi(\mathbf{x})$ are derived by neural networks. The architecture of VAE is shown as Fig 2.

SGVB [16, 30] is a variational inference algorithm that is often used along with VAE, where the approximated posterior and the generative model are jointly trained by maximizing the evidence lower bound (ELBO, Eqn (1)). We did not adopt more advanced variational inference algorithms, since SGVB already works.

$$\begin{aligned}\log p_\theta(\mathbf{x}) &\geq \log p_\theta(\mathbf{x}) - \mathrm{KL}\left[q_\phi(\mathbf{z}|\mathbf{x}) \middle\| p_\theta(\mathbf{z}|\mathbf{x})\right] \\ &= \mathcal{L}(\mathbf{x}) \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}) + \log p_\theta(\mathbf{z}|\mathbf{x}) - \log q_\phi(\mathbf{z}|\mathbf{x})\right] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})\right] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z}) + \log p_\theta(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})\right]\end{aligned} \tag{1}$$

Monte Carlo integration [10] is often adopted to approximate the expectation in Eqn (1), as Eqn (2), where $\mathbf{z}^{(l)}, l = 1 \ldots L$ are samples from $q_\phi(\mathbf{z}|\mathbf{x})$. We stick to this method throughout this paper.

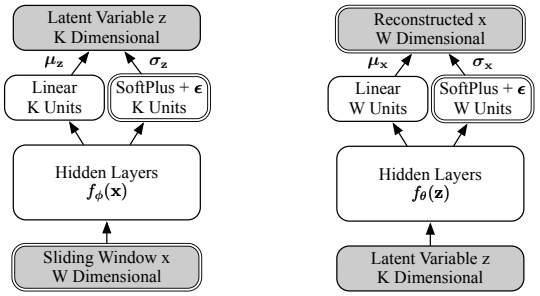
$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[f(\mathbf{z})] \approx \frac{1}{L}\sum_{l=1}^{L} f(\mathbf{z}^{(l)}) \tag{2}$$

3 ARCHITECTURE

The overall architecture of our algorithm *Donut* is illustrated as Fig 3. The three key techniques are *Modified ELBO* and *Missing Data Injection* during training, and *MCMC Imputation* in detection.

3.1 Network Structure

As aforementioned in § 2.1, the KPIs studied in this paper are assumed to be time sequences with Gaussian noises. However, VAE is not a sequential model, thus we apply sliding windows [31] of length W over the KPIs: for each point x_t, we use $x_{t-W+1}, \ldots, x_t$ as the $\mathbf{x}$ vector of VAE. This sliding window was first adopted because of its simplicity, but it turns out to actually bring an important and beneficial consequence, which will be discussed in § 5.1.



(a) Variational net $q_\phi(\mathbf{z}|\mathbf{x})$ **(b) Generative net $p_\theta(\mathbf{x}|\mathbf{z})$**

Figure 4: Network structure of *Donut*. Gray nodes are random variables, and white nodes are layers. The double lines highlight our special designs upon a general VAE.

The overall network structure of *Donut* is illustrated in Fig 4, where the components with double-lined outlines (*e.g.*, Sliding Window x, W Dimensional at bottom left) are our new designs and the remaining components are from standard VAEs. The prior $p_\theta(\mathbf{z})$ is chosen to be $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Both $\mathbf{x}$ and $\mathbf{z}$ posterior are chosen to be diagonal Gaussian: $p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}_\mathbf{x}, {\boldsymbol{\sigma}_\mathbf{x}}^2\mathbf{I})$, and $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_\mathbf{z}, {\boldsymbol{\sigma}_\mathbf{z}}^2\mathbf{I})$, where $\boldsymbol{\mu}_\mathbf{x}$, $\boldsymbol{\mu}_\mathbf{z}$ and $\boldsymbol{\sigma}_\mathbf{x}$, $\boldsymbol{\sigma}_\mathbf{z}$ are the means and standard deviations of each independent Gaussian component. $\mathbf{z}$ is chosen to be K dimensional. Hidden features are extracted from $\mathbf{x}$ and $\mathbf{z}$, by separated hidden layers $f_\phi(\mathbf{x})$ and $f_\theta(\mathbf{z})$. Gaussian parameters of $\mathbf{x}$ and $\mathbf{z}$ are then derived from the hidden features. The means are derived from linear layers: $\boldsymbol{\mu}_\mathbf{x} = \mathbf{W}^\top_{\boldsymbol{\mu}_\mathbf{x}} f_\theta(\mathbf{z}) + \mathbf{b}_{\boldsymbol{\mu}_\mathbf{x}}$ and $\boldsymbol{\mu}_\mathbf{z} = \mathbf{W}^\top_{\boldsymbol{\mu}_\mathbf{z}} f_\phi(\mathbf{x}) + \mathbf{b}_{\boldsymbol{\mu}_\mathbf{z}}$. The standard deviations are derived from soft-plus layers, plus a non-negative small number $\boldsymbol{\epsilon}$: $\boldsymbol{\sigma}_\mathbf{x} = \mathrm{SoftPlus}[\mathbf{W}^\top_{\boldsymbol{\sigma}_\mathbf{x}} f_\theta(\mathbf{z}) + \mathbf{b}_{\boldsymbol{\sigma}_\mathbf{x}}] + \boldsymbol{\epsilon}$ and $\boldsymbol{\sigma}_\mathbf{z} = \mathrm{SoftPlus}[\mathbf{W}^\top_{\boldsymbol{\sigma}_\mathbf{z}} f_\phi(\mathbf{x}) + \mathbf{b}_{\boldsymbol{\sigma}_\mathbf{z}}] + \boldsymbol{\epsilon}$, where $\mathrm{SoftPlus}[a] = \log[\exp(a) + 1]$. All the $\mathbf{W}$-s and $\mathbf{b}$-s presented here are parameters of corresponding layers. Note when scalar function $f(x)$ is applied on vector $\mathbf{x}$, it means to apply on every component.

We choose to derive $\boldsymbol{\sigma}_\mathbf{x}$ and $\boldsymbol{\sigma}_\mathbf{z}$ in such a way, instead of deriving $\log \boldsymbol{\sigma}_\mathbf{x}$ and $\log \boldsymbol{\sigma}_\mathbf{z}$ using linear layers as others do, for the following reason. The local variations in the KPIs of our interest are so small that $\boldsymbol{\sigma}_\mathbf{x}$ and $\boldsymbol{\sigma}_\mathbf{z}$ would probably get extremely close to zero, making $\log \boldsymbol{\sigma}_\mathbf{x}$ and $\log \boldsymbol{\sigma}_\mathbf{z}$ unbounded. This would cause severe numerical problems when computing the likelihoods of Gaussian variables. We thus use the soft-plus and the $\boldsymbol{\epsilon}$ trick to prevent such problems.

3.2 Training

Training is straightforward by optimizing the ELBO (Eqn (1)) with SGVB [16] algorithm. Since it is reported by [16] that one sample is sufficient for computing the ELBO when training VAE with the SGVB algorithm, we let sampling number $L = 1$ during training. The windows of $\mathbf{x}$ are randomly shuffled before every epoch, which is beneficial for stochastic gradient descent. A sufficiently large number of $\mathbf{x}$ are taken in every mini-batch, which is critical for stabilizing the training, since sampling introduces extra randomness.

As discussed in § 2.2, the VAE based anomaly detection works by learning normal patterns, thus we need to avoid learning abnormal patterns whenever possible. Note that the "anomalies" in training are labeled anomalies, and there can be no labels for a given KPI, in which case the anomaly detection becomes an unsupervised one.

One might be tempted to replace labeled anomalies (if any) and missing points (known) in training data with synthetic values. Some

previous work has proposed methods to impute missing data, *e.g.*, [34], but it is hard to produce data that follow the "normal patterns" well enough. More importantly, training a generative model with data generated by another algorithm is quite absurd, since one major application of generative models is exactly to generate data. Using data imputed by any algorithm weaker than VAE would potentially downgrade the performance. Thus we do not adopt missing data imputation before training VAE, instead we choose to simply fill the missing points as zeros (in the *Data Preparation* step in Fig 3), and then modify the ELBO to exclude the contribution of anomalies and missing points (shown as Modified ELBO (**M-ELBO** for short hereafter) in the *Training* step in Fig 3).

More specifically, we modify the standard ELBO in Eqn (1) to our version Eqn (3). α_w is defined as an indicator, where $a_w = 1$ indicates x_w being not anomaly or missing, and $a_w = 0$ otherwise. β is defined as $(\sum_{w=1}^{W} \alpha_w)/W$. Note that Eqn (3) still holds when there is no labeled anomalies in the training data. The contribution of $p_\theta(x_w|\mathbf{z})$ from labeled anomalies and missing points are directly excluded by α_w, while the scaling factor β shrinks the contribution of $p_\theta(\mathbf{z})$ according to the ratio of normal points in $\mathbf{x}$. This modification trains *Donut* to correctly reconstruct the normal points within $\mathbf{x}$, even if some points in $\mathbf{x}$ are abnormal. We do not shrink $q_\phi(\mathbf{z}|\mathbf{x})$, because of the following two considerations. Unlike $p_\theta(\mathbf{z})$, which is part of the generative network (*i.e.*, model of the "normal patterns"), $q_\phi(\mathbf{z}|\mathbf{x})$ just describes the mapping from $\mathbf{x}$ to $\mathbf{z}$, without considering"normal patterns". Thus, discounting the contribution of $q_\phi(\mathbf{z}|\mathbf{x})$ seems not necessary. Another reason is that $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[-\log q_\phi(\mathbf{z}|\mathbf{x})]$ is exactly the entropy of $q_\phi(\mathbf{z}|\mathbf{x})$. This entropy term actually has some other roles in training (which will be discussed in § 5.3), thus might be better kept untouched.

$$\widetilde{\mathcal{L}}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\sum_{w=1}^{W} \alpha_w \log p_\theta(x_w|\mathbf{z}) + \beta \log p_\theta(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})\right] \quad (3)$$

Besides Eqn (3), another way to deal with anomalies and missing points is to exclude all windows containing these points from training data. This approach turns out to be inferior to M-ELBO. We will demonstrate the performance of both approaches in § 4.5.

Furthermore, we also introduce missing data injection in training: we randomly set λ ratio of normal points to be zero, as if they are missing points. With more missing points, *Donut* is trained more often to reconstruct normal points when given abnormal $\mathbf{x}$, thus the effect of M-ELBO is amplified. This injection is done before every epoch, and the points are recovered once the epoch is finished. This missing data injection is shown in the *Training* step in Fig 3.

3.3 Detection

Generative models like VAE can derive various outputs. In the scope of anomaly detection, the likelihood of observation window $\mathbf{x}$, *i.e.*, $p_\theta(\mathbf{x})$ in VAE, is an important output, since we want to see how well a given $\mathbf{x}$ follows the normal patterns. Monte Carlo methods can be adopted to compute the probability density of $\mathbf{x}$, by $p_\theta(\mathbf{x}) = \mathbb{E}_{p_\theta(\mathbf{z})}[p_\theta(\mathbf{x}|\mathbf{z})]$. Despite the theoretically nice interpretation, sampling on the prior actually does not work well enough in practice, as will be shown in § 4.

Instead of sampling on the prior, one may seek to derive useful outputs with the variational posterior $q_\phi(\mathbf{z}|\mathbf{x})$. One choice is to compute $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[p_\theta(\mathbf{x}|\mathbf{z})]$. Although similar to $p_\theta(\mathbf{x})$, it is not

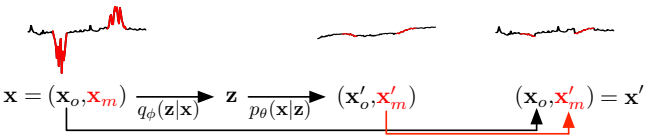


Figure 5: Illustration of one iteration in MCMC. $\mathbf{x}$ is decomposed as $(\mathbf{x}_o, \mathbf{x}_m)$, then $\mathbf{x}_o$ is fixed and $\mathbf{x}_m$ is replaced by $\mathbf{x}'_m$ from the reconstruction sample, in order to get the new $\mathbf{x}'$.

a well-defined probability density. Another choice is to compute $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]$, which is adopted in [2], named as "reconstruction probability". These two choices are very similar. Since only the ordering rather than the exact values of anomaly scores are concerned in anomaly detection, we follow [2] and use the latter one. As an alternative, the ELBO (Eqn (1)) may also be used for approximating $\log p_\theta(\mathbf{x})$, as in [33]. However, the extra term $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})\right]$ in ELBO makes its internal mechanism hard to understand. Since the experiments in [33] does not support this alternative's superiority, we choose not to use it.

During detection, the anomalies and missing points in a testing window $\mathbf{x}$ can bring bias to the mapped $\mathbf{z}$, and further make the reconstruction probability inaccurate, which would be discussed in § 5.2. Since the missing points are always known (as "null"), we have the chance to eliminate the biases introduced by missing points. We choose to adopt the MCMC-based missing data imputation technique with the trained VAE, which is proposed by [30]. Meanwhile, we do not know the exact positions of anomalies before detection, thus MCMC cannot be adopted on anomalies.

More specifically, the testing $\mathbf{x}$ is divided into observed and missing parts, *i.e.*, $(\mathbf{x}_o, \mathbf{x}_m)$. A $\mathbf{z}$ sample is obtained from $q_\phi(\mathbf{z}|\mathbf{x}_o, \mathbf{x}_m)$, then a reconstruction sample $(\mathbf{x}'_o, \mathbf{x}'_m)$ is obtained from $p_\theta(\mathbf{x}_o, \mathbf{x}_m|\mathbf{z})$. $(\mathbf{x}_o, \mathbf{x}_m)$ is then replaced by $(\mathbf{x}_o, \mathbf{x}'_m)$, *i.e.*, the observed points are fixed and the missing points are set to new values. This process is iterated for M times, then the final $(\mathbf{x}_o, \mathbf{x}'_m)$ is used for computing the reconstruction probability. The intermediate $\mathbf{x}'_m$ will keep getting closer to normal values during the whole procedure. Given sufficiently large M, the biases can be reduced, and we can get a more accurate reconstruction probability. The MCMC method is illustrated in Fig 5 and is shown in the *Detection* step in Fig 3.

After MCMC, we take L samples of $\mathbf{z}$ to compute the reconstruction probability by Monte Carlo integration. Although we may compute the reconstruction probability for each point in every window of $\mathbf{x}$, we only use the score for the last point (*i.e.*, x_t in $x_{t-T+1}, \ldots, x_t$), since we want to respond to anomalies as soon as possible during detection. We will still use vector notations in later texts, corresponding to the architecture of VAE.

4 EVALUATION

4.1 Datasets

We obtain 18 well-maintained business KPIs (where the time span is long enough for training and evaluation) from a large Internet company. All KPIs have an interval of 1 minute between two observations. We choose 3 datasets, denoted as $\mathcal{A}$, $\mathcal{B}$ and C, which have relatively small, medium and large noises among the 18 datasets, so we can evaluate *Donut* for noises at different levels. We divide each dataset into training, validation and testing sets, whose ratios are 49%, 21%, 30% respectively. Figures of datasets $\mathcal{A}$, $\mathcal{B}$ and C are

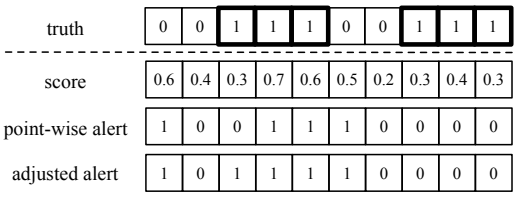


Figure 6: Illustration of the strategy for modified metrics. The first row is the truth with 10 contiguous points and two anomaly segments highlighted in the shaded squares. The detector scores are shown in the second row. The third row shows the point-wise detector results with a threshold of 0.5. The forth row shows the detector results after adjustment. We shall get precision 0.6, and recall 0.5. From the third row, the alert delay for the first segment is 1 interval (1 minute).

shown in Fig 1, while statistics are shown in Table 1. The operators of the Internet company labeled all the anomalies in these three datasets. For evaluation purpose, we can consider we have the ground truth of all anomalies in these three datasets.

DataSet	$\mathcal{A}$	$\mathcal{B}$	C
Total points	296460	317522	285120
Missing points	1222/0.41%	1117/0.35%	304/0.11%
Anomaly points	1213/0.41%	1883/0.59%	4394/1.54%
Total windows*	296341	317403	285001
Abnormal windows**	20460/6.90%	20747/6.54%	17288/6.07%

* Each sliding window has a length $W = 120$.
** Each abnormal window contains at least one anomaly or missing point.

Table 1: Statistics of $\mathcal{A}$, $\mathcal{B}$ and C.

4.2 Performance Metrics

In our evaluation, we totally ignore outputs of all algorithms at missing points ("null") since they are straightforward to identify.

All the algorithms evaluated in this paper compute one anomaly score for each point. A threshold can be chosen to do the decision: if the score for a point is greater than the threshold, an alert should be triggered. In this way, anomaly detection is similar to a classification problem, and we may compute the precision and recall corresponding to each threshold. We may further compute the AUC, which is the average precision over recalls, given all possible thresholds; or the F-score, which is the harmonic mean of precision and recall, given one particular threshold. We may also enumerate all thresholds, obtaining all F-scores, and use the *best F-score* as the metric. The best F-score indicates the best possible performance of a model on a particular testing set, given an optimal global threshold. In practice, the best F-score is mostly consistent with AUC, except for slight differences (see Fig 7). We prefer the best F-score to AUC, since it should be more important to have an excellent F-score at a certain threshold than to have just high but not so excellent F-scores on most thresholds.

In real applications, the human operators generally do not care about the point-wise metrics. It is acceptable for an algorithm to trigger an alert for any point in a contiguous anomaly segment, if the delay is not too long. Some metrics for anomaly detection have been proposed to accommodate this preference, *e.g.*, [20], but most are not widely accepted, likely because they are too complicated. We instead use a simple strategy: if any point in an anomaly segment in the ground truth can be detected by a chosen threshold, we say

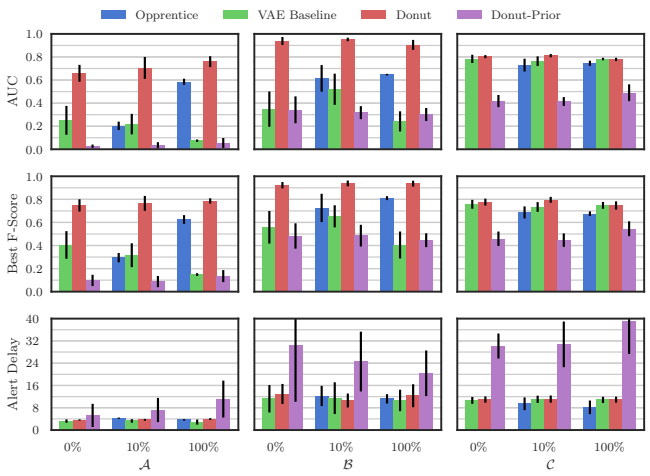


Figure 7: AUC, the best F-Score, and the average alert delay corresponding to the best F-score. $\mathcal{A}$, $\mathcal{B}$ and C are the three datasets. "0%", "10%" and "100%" are the ratio of the labels preserved in training. Note there is no result for Opprentice when there are 0% of anomaly labels. The black stick on top of each bar is the deviation of 10 repeated experiments.

this segment is detected correctly, and all points in this segment are treated as if they can be detected by this threshold. Meanwhile, the points outside the anomaly segments are treated as usual. The precision, recall, AUC, F-score and best F-score are then computed accordingly. This approach is illustrated in Fig 6.

In addition to the accuracy metric, we compute the alert delay for each detected segment, which is also important to the operators. For a true positive segment, the alert delay is the time difference between the first point and the first detected point in the segment.

4.3 Experiment Setup

We set the window size W to be 120, which spans 2 hours in our datasets. The choice of W is restricted by two factors. On the one hand, too small a W will cause the model to be unable to capture the patterns, since the model is expected to recognize what the normal pattern is with the information only from the window (see § 5.1). On the other hand, too large a W will increase the risk of over-fitting, since we stick to fully-connected layers without weight sharing, thus the number of model parameters is proportional to W. We set the latent dimension K to be 3 for $\mathcal{B}$ and C, since the 3-d dimensional space can be easily visualized for analysis and luckily $K = 3$ works well empirically for for $\mathcal{B}$ and C. As for $\mathcal{A}$, we found 3 is too small, so we empirically increase K to 8. These empirical choices of K are proven to be quite good on testing set, as will be shown in Fig 9. The hidden layers of $q_\phi(\mathbf{z}|\mathbf{x})$ and $p_\theta(\mathbf{x}|\mathbf{z})$ are both chosen as two ReLU layers, each with 100 units, which makes the variational and generative network have equal size. We did not carry out exhaustive search on the structure of hidden networks.

Other hyper-parameters are also chosen empirically. We use 10^{-4} as ϵ of the std layer. We use 0.01 as the injection ratio λ. We use 10 as the MCMC iteration count M, and use 1024 as the sampling number L of Monte Carlo integration. We use 256 as the batch size for training, and run for 250 epochs. We use Adam optimizer [15], with an initial learning rate of 10^{-3}. We discount the learning rate

by 0.75 after every 10 epochs. We apply L2 regularization to the hidden layers, with a coefficient of 10^{-3}. We clip the gradients by norm, with a limit of 10.0.

In order to evaluate *Donut* with no labels, we ignore all the labels. For the case of occasional labels, we down-sample the anomaly labels of training and validation set to make it contain 10% of labeled anomalies. Note that missing points are not down-sampled. We keep throwing away anomaly segments randomly, with a probability that is proportional to the length of each segment, until the desired down-sampling rate is reached. We use this approach instead of randomly throwing away individual anomaly points, because KPIs are time sequences and each anomaly point could leak information about its neighboring points, resulting in over-estimated performance. Such downsampling are done 10 times, which enables us to do 10 independent, repeated experiments. Overall for each dataset, we have three versions: 0% labels, 10% labels, and 100% labels.

4.4 Overall Performance

We measure the AUC, the best F-Score, and the average alert delay corresponding to the best F-score in Fig 7 of *Donut*, and compared with three selected algorithms.

Opprentice [23] is an ensemble supervised framework using Random Forest classifier. On datasets similar to ours, Opprentice is reported to consistently and significantly outperform 14 anomaly detectors based on traditional statistical models (*e.g.*, [6, 17, 18, 22, 24, 25, 29, 37, 38]), with in total 133 enumerated configurations of hyper-parameters for these detectors. Thus, in our evaluation of *Donut*, Opprentice not only serves as a state-of-art competitor algorithm from the non deep learning areas, but also serves as a proxy to compare with the empirical performance "upper bound" of these traditional anomaly detectors.

VAE baseline. The VAE-based anomaly detection in [2] does not deal with time sequences, thus we set up the VAE baseline as follows. First, the VAE baseline has the same network structure as *Donut*, as shown in Fig 4. Second, among all the techniques in Fig 3, only those techniques in the Data Preparation step are used. Third, as suggested by [2], we exclude **all windows** containing either labeled anomalies or missing points from training data.

Donut-Prior. Given that a generative model learns $p(\mathbf{x})$ by nature, while in VAE $p(\mathbf{x})$ is defined as $\mathbb{E}_{p_\theta(\mathbf{z})}[p_\theta(\mathbf{x}|\mathbf{z})]$, we also evaluate the prior counterpart of reconstruction probability, *i.e.*, $\mathbb{E}_{p_\theta(\mathbf{z})}[\log p_\theta(\mathbf{x}|\mathbf{z})]$. We just need a baseline of the prior, so we compute the prior expectation by plain Monte Carlo integration, without advanced techniques to improve the result.

The best F-score of *Donut* is quite satisfactory in totally unsupervised case, ranges from 0.75 to 0.9, better than the supervised Opprentice in all cases. In fact, when labels are incomplete, the best F-score of the Opprentice drops heavily in $\mathcal{A}$ and $\mathcal{B}$, only remaining acceptable in C. The number of anomalies are much larger in C than $\mathcal{A}$ and $\mathcal{B}$, while having 10% of labels are likely to be just enough for training. *Donut* has an outstanding performance in the unsupervised scenario, and we see that feeding anomaly labels into *Donut* would in general make it work even better. There is, however, an unusual behavior of *Donut*, where the best F-score in C, as well as the AUC in $\mathcal{B}$ and C, are slightly worse with 100% labels than 10%. This is likely an optimization problem, where the unlabeled anomalies might cause training to be unstable, accidentally pulling

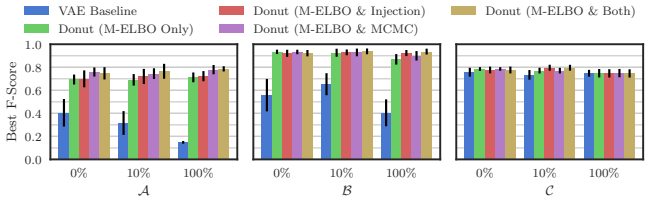


Figure 8: Best F-score of (1) VAE baseline, (2) *Donut* with M-ELBO, (3) M-ELBO + missing data injection, (4) M-ELBO + MCMC, and (5) M-ELBO + both MCMC and injection. The M-ELBO alone contributes most of the improvement.

the model out of a sub-optimal equilibrium (§ 5.4). Such phenomenon seems to diminish when K increases from 3 ($\mathcal{B}$ and C) to 8 ($\mathcal{A}$). Fortunately, it does not matter too much, so we would suggest to use labels in *Donut* whenever possible.

Donut outperforms the VAE baseline by a large margin in $\mathcal{A}$ and $\mathcal{B}$, while it does not show such great advantage in C. In fact, the relative advantage of *Donut* is the largest in $\mathcal{A}$, medium in $\mathcal{B}$, and the smallest in C. This is caused by the following reasons. Naturally, VAE models normal $\mathbf{x}$. As a result, the reconstruction probability actually expects $\mathbf{x}$ to be mostly normal (see § 5.1). However, since $\mathbf{x}$ are sliding windows of KPIs and we are required to produce one anomaly score for every point, it is sometimes inevitable to have abnormal points in $\mathbf{x}$. This causes the VAE baseline to suffer a lot. In contrast, the techniques developed in this paper enhances the ability of *Donut* to produce reliable outputs even when anomalies present in earlier points in the same window. Meanwhile, abnormal points with similar abnormal magnitude would appear relatively "more abnormal" when the KPI is smoother. Given that $\mathcal{A}$ is the smoothest, $\mathcal{B}$ is medium, and C is the least smoothest, above observation in the relative advantage is not surprising.

Finally, the best F-score of the *Donut-Prior* is much worse than the reconstruction probability, especially when the dimension of $\mathbf{z}$ is larger. However, it is worth mentioning that the posterior expectation in reconstruction probability only works under certain conditions (§ 5.2). Fortunately, this problem does not matter too much to *Donut* (see § 5.2). As such, the reconstruction probability can be used without too much concern.

The average alert delays of *Donut*, Opprentice and VAE Baseline are acceptable over all datasets, whereas Donut-Prior is not. Meanwhile, the best F-score of *Donut* is much better than others. In conclusion, *Donut* could achieve the best performance without increasing the alert delay, thus *Donut* is practical for operators.

4.5 Effects of *Donut* Techniques

We have proposed three techniques in this paper: (1) M-ELBO (Eqn (3)), (2) missing data injection, and (3) MCMC imputation. In Fig 9, we present the best F-score of *Donut* with four possible combinations of these techniques, plus the VAE baseline for comparison. These techniques are closely related to the KDE interpretation, which will be discussed further in § 5.2.

M-ELBO alone contributes most of the improvement over the VAE baseline. It works by training *Donut* to get used to possible abnormal points in $\mathbf{x}$, and to produce desired outputs in such cases. Although we expected M-ELBO to work, we did not expect it to work such well. In conclusion, **it would not be a good practice to**

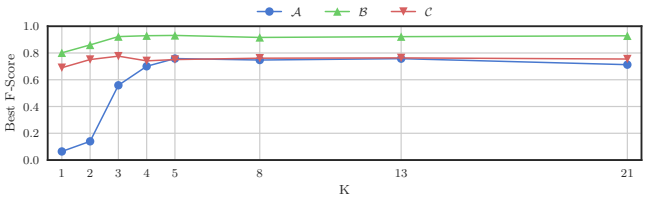


Figure 9: The best F-score of unsupervised *Donut* with different K, averaged over 10 repeated experiments.

train a VAE for anomaly detection using only normal data, although it seems natural for a generative model (§ 5.2). To the best of our knowledge, M-ELBO and its importance have never been stated in previous work, thus is a major contribution of ours.

Missing data injection is designed for amplifying the effect of M-ELBO, and can actually be seen as a data augmentation method. In fact, it would be better if we inject not only missing points, but also synthetically generated anomalies during training. However, it is difficult to generate anomalies similar enough to the real ones, which should be a large topic and is out of the scope of this paper. We thus only inject the missing points. The improvement of best F-score introduced by missing data injection is not very significant, and in the case of 0% labels on $\mathcal{B}$ and C, it is slightly worse than M-ELBO only. This is likely because the injection introduces extra randomness to training, such that it demands larger training epochs, compared to the case of M-ELBO only. We are not sure how many number of epochs to run when the injection is adopted, in order to get an objective comparison, thus we just use the same epochs in all cases, leaving the result as it is. We still recommend to use missing data injection, even with a cost of larger training epochs, as it is expected to work with a large chance.

MCMC imputation is also designed to help *Donut* deal with abnormal points. Although *Donut* obtains significant improvement of best F-score with MCMC in only some cases, it never harms the performance. According to [30], this should be an expected result. We thus recommend to always adopt MCMC in detection.

In conclusion, we recommend to use all the three techniques of *Donut*. The result of such configuration is also presented in Fig 8.

4.6 Impact of K

The number of z dimensions, *i.e.*, K, plays an important role. Too small a K would potentially cause under-fitting, or sub-optimal equilibrium (see § 5.4). On the other hand, too large a K would probably cause the reconstruction probability unable to find a good posterior (see § 5.1). It is difficult to choose a good K in totally unsupervised scenario, thus we leave it as a future work.

In Fig 9, we present the average best F-score with different K on testing set for unsupervised *Donut*. This does not help us choose the best K (since we cannot use testing test to pick K), but can show our empirical choice of 8, 3, 3 is quite good. The best F-score reaches maximum at 5 for $\mathcal{A}$, 4 for $\mathcal{B}$ and 3 for C. In other words, the best F-score could be achieved with fairly small K. On the other hand, the best F-score does not drop too heavily for K up to 21. This gives us a large room to empirically choose K. Finally, we notice that smoother KPIs seem to demand larger K. Such phenomenon is not fully studied in this paper, and we leave it as a future work. Based on the observations in Fig 9, for KPIs similar to $\mathcal{A}$, $\mathcal{B}$ or C, we suggest an empirical choice of K within the range from 5 to 10.

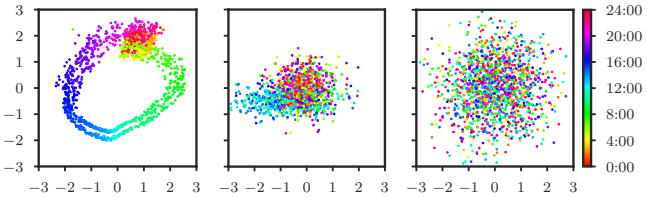


Figure 10: The z layout of dataset $\mathcal{B}$ with (a) *Donut*, (b) untrained VAE, (c) VAE trained using $\mathbb{E}[\log p_\theta(\mathbf{z})]+\mathrm{H}[\mathbf{z}|\mathbf{x}]$ as loss. Figures are plotted by sampling z from $q_\phi(\mathbf{z}|\mathbf{x})$, corresponding to normal x randomly chosen from the testing set. K is chosen as 2, so the x- and y-axis are the two dimensions of z samples. We plot z samples instead of $\boldsymbol{\mu}_\mathbf{z}$ of $q_\phi(\mathbf{z}|\mathbf{x})$, since we want to take into account the effects of $\sigma_\mathbf{z}$ in the figures. The color of z a sample denotes its time of the day.

5 ANALYSIS

5.1 KDE Interpretation

Although the reconstruction probability $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]$ has been adopted in [2, 33], how it actually works has not yet been made clear. Some may see it as a variant of $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[p_\theta(\mathbf{x}|\mathbf{z})]$, but $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[p_\theta(\mathbf{x}|\mathbf{z})] = \int p_\theta(\mathbf{x}|\mathbf{z})q_\phi(\mathbf{z}|\mathbf{x})\mathrm{d}\mathbf{z}$, which is definitely not a well-defined probability[3]. Thus neither of [2, 33] can be explained by the probabilistic framework. We hereby propose the KDE (kernel density estimation) interpretation for the reconstruction probability, and for the entire *Donut* algorithm.

The posterior $q_\phi(\mathbf{z}|\mathbf{x})$ for normal $\mathbf{x}$ exhibits time gradient, as Fig 10a shows. The windows of $\mathbf{x}$ at contiguous time (**contiguous** $\mathbf{x}$ for short hereafter) are mapped to nearby $q_\phi(\mathbf{z}|\mathbf{x})$, mostly with small variance $\boldsymbol{\sigma}_\mathbf{z}$ (see Fig 11). The $q_\phi(\mathbf{z}|\mathbf{x})$ are thus organized in smooth transition, causing $\mathbf{z}$ samples to exhibit color gradient in the figure. We name this structure "time gradient". The KPIs in this paper are smooth in general, so contiguous $\mathbf{x}$ are highly similar. The root cause of time gradient is the transition of $q_\phi(\mathbf{z}|\mathbf{x})$ in the shape of $\mathbf{x}$ (rather than the one in time), because *Donut* consumes only the shape of $\mathbf{x}$ and no time information. Time gradient benefits the generalization of *Donut* on unseen data: if we have a posterior $q_\phi(\mathbf{z}|\mathbf{x})$ somewhere between two training posteriors, it would be well-defined, avoiding absurd detection output.

For a partially abnormal $\mathbf{x}$[4], the dimension reduction would allow *Donut* to recognize its normal pattern $\tilde{\mathbf{x}}$, and cause $q_\phi(\mathbf{z}|\mathbf{x})$ to be approximately $q_\phi(\mathbf{z}|\tilde{\mathbf{x}})$. This effect is caused by the following reasons. *Donut* is trained to reconstruct normal points in training samples with best efforts, while the dimension reduction causes *Donut* to be only able to capture a small amount of information from $\mathbf{x}$. As a result, only the overall shape is encoded in $q_\phi(\mathbf{z}|\mathbf{x})$. The abnormal information is likely to be dropped in this procedure. However, if a $\mathbf{x}$ is too abnormal, *Donut* might fail to recognize any normal $\tilde{\mathbf{x}}$, such that $q_\phi(\mathbf{z}|\mathbf{x})$ would become ill-defined.

The fact that $q_\phi(\mathbf{z}|\mathbf{x})$ for a partially abnormal $\mathbf{x}$ would be similar to $q_\phi(\mathbf{z}|\tilde{\mathbf{x}})$ brings special meanings to the reconstruction probability

[3] In general it should give no useful information by computing the expectation of $\log p_\theta(\mathbf{x}|\mathbf{z})$ upon the posterior $q_\phi(\mathbf{z}|\mathbf{x})$, using a **potentially abnormal x**.

[4] We call a $\mathbf{x}$ partially abnormal if only a small portion of points within $\mathbf{x}$ are abnormal, such that we can easily tell what normal pattern $\mathbf{x}$ should follow.

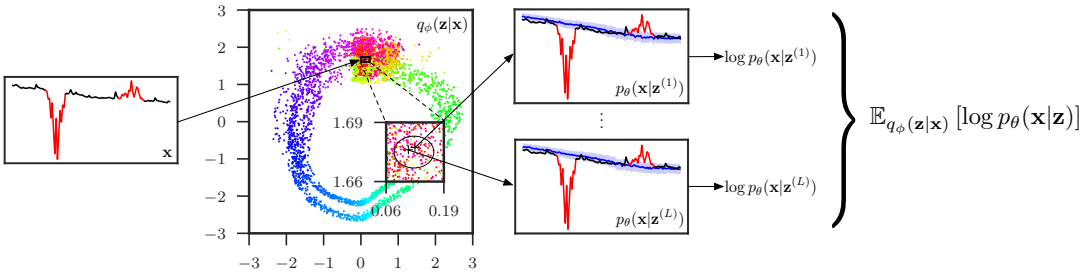


Figure 11: Illustration of the KDE interpretation. For a given $\mathbf{x}$ potentially with anomalies, *Donut* tries to recognize what normal pattern it follows, encoded as $q_\phi(\mathbf{z}|\mathbf{x})$. The black ellipse in the middle figure denotes the 3-$\sigma_\mathbf{z}$ region of $q_\phi(\mathbf{z}|\mathbf{x})$. L samples of $\mathbf{z}$ are then taken from $q_\phi(\mathbf{z}|\mathbf{x})$, denoted as the crosses in the middle figure. Each $\mathbf{z}$ is associated with a density estimator kernel $\log p_\theta(\mathbf{x}|\mathbf{z})$. The blue curves in the right two figures are $\mu_\mathbf{x}$ of each kernel, while the surrounding stripes are $\sigma_\mathbf{x}$. Finally, the values of $\log p_\theta(\mathbf{x}|\mathbf{z})$ are computed from each kernel, and further averaged together as the reconstruction probability.

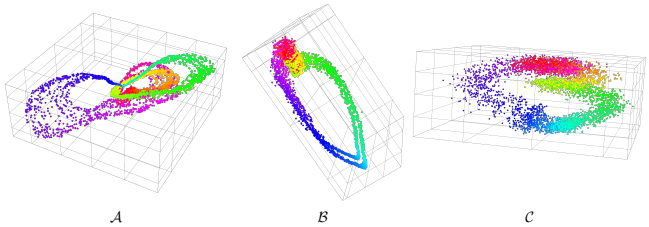


Figure 12: 3-d latent space of all three datasets.

in *Donut*. Since M-ELBO is maximized with regard to normal patterns during training, $\log p_\theta(\mathbf{x}|\mathbf{z})$ for $\mathbf{z} \sim q_\phi(\mathbf{z}|\tilde{\mathbf{x}})$ should produce high scores for $\mathbf{x}$ similar to $\tilde{\mathbf{x}}$, and vise versa. That is to say, each $\log p_\theta(\mathbf{x}|\mathbf{z})$ can be used as a density estimator, indicating how well $\mathbf{x}$ follows the normal pattern $\tilde{\mathbf{x}}$. The posterior expectation then sums up the scores from all $\log p_\theta(\mathbf{x}|\mathbf{z})$, with the weight $q_\phi(\mathbf{z}|\mathbf{x})$ for each $\mathbf{z}$. This procedure is very similar to weighted kernel density estimation [11, 14]. We thus carry out the KDE interpretation: **the reconstruction probability $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]$ in *Donut* can be seen as weighted kernel density estimation, with $q_\phi(\mathbf{z}|\mathbf{x})$ as weights and $\log p_\theta(\mathbf{x}|\mathbf{z})$ as kernels** [5].

Fig 11 is an illustration of the KDE interpretation. We also visualize the 3-d latent spaces of all datasets in Fig 12. From the KDE interpretation, we suspect the prior expectation would not work well, whatever technique is adopted to improve the result: sampling on the prior should obtain kernels for all patterns of $\mathbf{x}$, potentially confusing the density estimation for a particular $\mathbf{x}$.

5.2 Find Good Posteriors for Abnormal x

Donut can recognize the normal pattern of a partially abnormal $\mathbf{x}$, and find a good posterior for estimating how well $\mathbf{x}$ follows the normal pattern. We now analyze how the techniques in *Donut* can enhance such ability of finding good posteriors.

Donut is forced to reconstruct normal points within abnormal windows correctly during training, by M-ELBO. It is thus explicitly trained to find good posteriors. This is the main reason why M-ELBO plays a vital role in Fig 8. Missing data injection amplifies the effect of M-ELBO, with synthetically generated missing points. On the other hand, MCMC imputation does not change the

[5]The weights $q_\phi(\mathbf{z}|\mathbf{x})$ are implicitly applied by sampling in Monte Carlo integration.

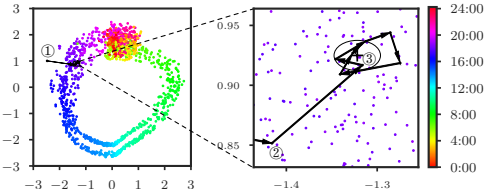


Figure 13: MCMC visualization. A normal $\mathbf{x}$ is chosen, whose posterior $q_\phi(\mathbf{z}|\mathbf{x})$ is plotted at right: the cross denotes $\mu_\mathbf{x}$ and the ellipse denotes its 3-$\sigma_\mathbf{x}$ region. We randomly set 15% $\mathbf{x}$ points as missing, to obtain the abnormal $\mathbf{x}'$. We run MCMC over $\mathbf{x}'$ with 10 iterations. At first, the $\mathbf{z}$ sample is far from $q_\phi(\mathbf{z}|\mathbf{x})$. After that, $\mathbf{z}$ samples quickly approach $q_\phi(\mathbf{z}|\mathbf{x})$, and begin to move around $q_\phi(\mathbf{z}|\mathbf{x})$ after only 3 iterations.

training process. Instead, it improves the detection, by iteratively approaching better posteriors, as illustrated in Fig 13.

Despite these techniques, *Donut* may still fail to find a good posterior, if there are too many anomalies in $\mathbf{x}$. In our scenario, the KPIs are time sequences, with one point per minute. For long-lasting anomalies, having the correct detection scores and raise alerts at first few minutes are sufficient in our context[6]. The operators can take action once any score reaches the threshold, and simply ignore the following inaccurate scores. **Nevertheless, the KDE interpretation can help us know the limitations of reconstruction probability, in order to use it properly.**

5.3 Causes of Time Gradient

In this section we discuss the causes of the time gradient effect. To simplify the discussion, let us assume training $\mathbf{x}$ are all normal, thus M-ELBO is now equivalent to the original ELBO. M-ELBO can then be decomposed into three terms as in Eqn (4) (we leave out some subscripts for shorter notation).

$$\begin{aligned}\mathcal{L}(\mathbf{x}) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z}) + \log p_\theta(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})\right] \\ &= \mathbb{E}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] + \mathbb{E}\left[\log p_\theta(\mathbf{z})\right] + \mathrm{H}\left[\mathbf{z}|\mathbf{x}\right] \qquad (4)\end{aligned}$$

The 1st term requires $\mathbf{z}$ samples from $q_\phi(\mathbf{z}|\mathbf{x})$ to have a high likelihood of reconstructing $\mathbf{x}$. As a result, $q_\phi(\mathbf{z}|\mathbf{x})$ for $\mathbf{x}$ with dissimilar shapes are separated. The 2nd term causes $q_\phi(\mathbf{z}|\mathbf{x})$ to concentrate on $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The 3rd term, the entropy of $q_\phi(\mathbf{z}|\mathbf{x})$, causes $q_\phi(\mathbf{z}|\mathbf{x})$

[6]In practice, ensuing and continuous alerts are typically filtered out anyway.

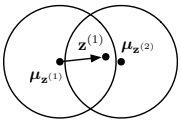


Figure 14: Suppose $\mu_{\mathbf{z}^{(1)}}$ and $\mu_{\mathbf{z}^{(2)}}$ are the mean of $q_\phi(\mathbf{z}|\mathbf{x})$ corresponding to training data $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$, with the surrounding circles represent $\sigma_{\mathbf{z}^{(1)}}$ and $\sigma_{\mathbf{z}^{(2)}}$. When these two distributions accidentally "overlaps" during training, the sample $\mathbf{z}^{(1)}$ from $q_\phi(\mathbf{z}|\mathbf{x}^{(1)})$ may get too close to $\mu_{\mathbf{z}^{(2)}}$, such that the reconstructed distribution will be close to $p_\theta(\mathbf{x}|\mathbf{z}^{(2)})$ with some $\mathbf{z}^{(2)}$ for $\mathbf{x}^{(2)}$. If $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ are dissimilar, $\log p_\theta(\mathbf{x}^{(1)}|\mathbf{z}^{(2)})$ in the loss will then effectively push $\mu_{\mathbf{z}^{(1)}}$ away from $\mu_{\mathbf{z}^{(2)}}$.

to expand wherever possible. Recall the 2nd term sets a restricted area for $q_\phi(\mathbf{z}|\mathbf{x})$ to expand (see Fig 10c for the combination effect of the 2nd and 3rd term). Taking the 1st term into account, this expansion would also stop if $q_\phi(\mathbf{z}|\mathbf{x})$ for two dissimilar $\mathbf{x}$ reach each other. In order for every $q_\phi(\mathbf{z}|\mathbf{x})$ to have a maximal territory when training converges (*i.e.*, these three terms reach an equilibrium), similar $\mathbf{x}$ would have to get close to each other, allowing $q_\phi(\mathbf{z}|\mathbf{x})$ to grow larger with overlapping boundaries. Since contiguous $\mathbf{x}$ are similar in seasonal KPIs (and vise versa), the time gradient would be a natural consequence, if such equilibrium could be achieved.

Next we discuss how the equilibrium could be achieved. The SGVB algorithm keeps pushing $q_\phi(\mathbf{z}|\mathbf{x})$ for dissimilar $\mathbf{x}$ away during training, as illustrated in Fig 14. The more dissimilar two $q_\phi(\mathbf{z}|\mathbf{x})$ are, the further they are pushed away. Since we initialize the variational network randomly, $q_\phi(\mathbf{z}|\mathbf{x})$ are mixed everywhere when training just begins, as Fig 10b shows. At this time, every $q_\phi(\mathbf{z}|\mathbf{x})$ are pushed away by all other $q_\phi(\mathbf{z}|\mathbf{x})$. Since $\mathbf{x}$ are sliding windows of KPIs, any pair of $\mathbf{x}$ far away in time will be generally more dissimilar, thus get pushed away further from each other. This gives $q_\phi(\mathbf{z}|\mathbf{x})$ an initial layout. As training goes on, the time gradient is fine-tuned and gradually established, as Fig 15a shows. The training dynamics also suggest that the learning rate annealing technique is very important, since it can gradually stabilize the layout.

Surprisingly, we cannot find any term in M-ELBO that directly pulls $q_\phi(\mathbf{z}|\mathbf{x})$ for similar $\mathbf{x}$ together. The time gradient is likely to be caused mainly by expansion ($\mathrm{H}[\mathbf{z}|\mathbf{x}]$), squeezing ($\mathbb{E}[\log p_\theta(\mathbf{z})]$), pushing ($\mathbb{E}[\log p_\theta(\mathbf{x}|\mathbf{z})]$), and the training dynamics (random initialization and SGVB). This could sometimes cause trouble, and result in sub-optimal layouts, as we shall see in § 5.4.

5.4 Sub-Optimal Equilibrium

$q_\phi(\mathbf{z}|\mathbf{x})$ may sometimes converge to a sub-optimal equilibrium. Fig 15b demonstrates such a problem, where the purple points accidentally get through the green points after the first 100 steps. The purple points push the green points away towards both sides, causing the green points to be totally cut off at around 5000 steps. As training goes on, the green points will be pushed even further, such that the model is locked to this sub-optimal equilibrium and never escapes. Such bad layout of $\mathbf{z}$ breaks the time gradient, where a testing $\mathbf{x}$ following green patterns might accidentally be mapped to somewhere between the green two halves and get recognized as purple. This would certainly downgrade the detection performance, according to the KDE interpretation.

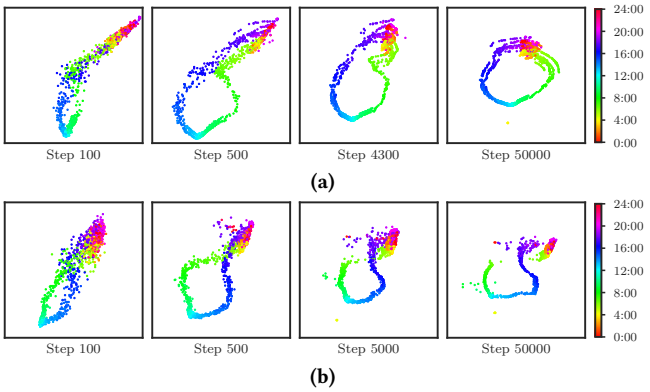


Figure 15: Evolution of the z space of dataset $\mathcal{B}$ during training. We sample normal x from validation set, and plot z samples accordingly. (a) converges to a good equilibrium, with a final F-score 0.871, while (b) converges to a sub-optimal one, with a final F-score 0.826. We plot step 4300 in (a), because it is a very important turning point, where the green points just begin to get away from the purple points.

When there are unlabeled anomalies, the training would become unstable so that the model might be accidentally brought out of a sub-optimal equilibrium and achieve a better equilibrium afterwards. With the help of early-stopping during training, the best encountered equilibrium is chosen eventually. This explains why sometimes having complete labels would not benefit the performance. This effect is likely to be less obvious with larger K, since having more dimensions gives $q_\phi(\mathbf{z}|\mathbf{x})$ extra freedom to grow, reducing the chance of bad layouts. When sub-optimal equilibrium is not a vital problem, the convergence of training then becomes more important, while having more labels definitely helps stabilize the training. In conclusion, using anomaly labels in *Donut* is likely to benefit the performance, as long as K is adequately large.

6 CONCLUSION

In this paper, we proposed an unsupervised anomaly detection algorithm *Donut* based on VAE for seasonal KPIs with local variations. The new techniques enabled *Donut* to greatly outperform state-of-art supervised and VAE-based anomaly detection algorithms. The best F-scores of *Donut* range from 0.75 to 0.90 for the studied KPIs.

Donut's excellent performance are explained by our theoretical analysis with KDE interpretation and the new discovery of the time gradient effect. Our experimental and theoretical analyses imply broader impacts: anomaly detection based on dimension reduction needs to use reconstruction; anomaly detection with generative models needs to train with both normal and abnormal data.

7 ACKNOWLEDGEMENTS

The work was supported by National Natural Science Foundation of China (NSFC) under grant No. 61472214 and No. 61472210, and Alibaba Innovative Research (AIR). We also thank Prof. Jun Zhu and his PhD. student Jiaxin Shi for helpful and constructive discussions.

REFERENCES

[1] Mennatallah Amer, Markus Goldstein, and Slim Abdennadher. 2013. Enhancing one-class support vector machines for unsupervised anomaly detection. In *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*. ACM, 8–15.

[2] Jinwon An and Sungzoon Cho. 2015. *Variational Autoencoder based Anomaly Detection using Reconstruction Probability.* Technical Report. SNU Data Mining Center. 1–18 pages.

[3] Matthew James Beal. 2003. *Variational algorithms for approximate Bayesian inference*. University of London London.

[4] Christopher M Bishop. 2006. *Pattern recognition and machine learning*. springer.

[5] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM computing surveys (CSUR)* 41, 3 (2009), 15.

[6] Yingying Chen, Ratul Mahajan, Baskar Sridharan, and Zhi-Li Zhang. 2013. A Provider-side View of Web Search Response Time. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM (SIGCOMM '13)*. ACM, New York, NY, USA, 243–254. https://doi.org/10.1145/2486001.2486035

[7] Sarah M Erfani, Sutharshan Rajasegarar, Shanika Karunasekera, and Christopher Leckie. 2016. High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. *Pattern Recognition* 58 (2016), 121–134.

[8] Romain Fontugne, Pierre Borgnat, Patrice Abry, and Kensuke Fukuda. 2010. MAWILab: Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking. In *Proceedings of the 6th International COnference (Co-NEXT '10)*. ACM, Article 8, 12 pages. https://doi.org/10.1145/1921168.1921179

[9] Zhouyu Fu, Weiming Hu, and Tieniu Tan. 2005. Similarity based vehicle trajectory clustering and anomaly detection. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, Vol. 2. IEEE, II–602.

[10] John Geweke. 1989. Bayesian inference in econometric models using Monte Carlo integration. *Econometrica: Journal of the Econometric Society* (1989), 1317–1339.

[11] Francisco J Goerlich Gisbert. 2003. Weighted samples, kernel density estimators and convergence. *Empirical Economics* 28, 2 (2003), 335–351.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[14] Wolfgang Härdle, Axel Werwatz, Marlene Müller, and Stefan Sperlich. 2004. Nonparametric density estimation. *Nonparametric and Semiparametric Models* (2004), 39–83.

[15] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[16] Diederik P Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *Proceedings of the International Conference on Learning Representations*.

[17] Florian Knorn and Douglas J Leith. 2008. Adaptive kalman filtering for anomaly detection in software appliances. In *INFOCOM Workshops 2008, IEEE*. IEEE, 1–6.

[18] Balachander Krishnamurthy, Subhabrata Sen, Yin Zhang, and Yan Chen. 2003. Sketch-based change detection: methods, evaluation, and applications. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*. ACM, 234–247.

[19] Nikolay Laptev, Saeed Amizadeh, and Ian Flint. 2015. Generic and scalable framework for automated time-series anomaly detection. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1939–1947.

[20] Alexander Lavin and Subutai Ahmad. 2015. Evaluating Real-Time Anomaly Detection Algorithms–The Numenta Anomaly Benchmark. In *Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on*. IEEE, 38–44.

[21] Rikard Laxhammar, Goran Falkman, and Egils Sviestins. 2009. Anomaly detection in sea traffic-a comparison of the gaussian mixture model and the kernel density estimator. In *Information Fusion, 2009. FUSION'09. 12th International Conference on*. IEEE, 756–763.

[22] Suk-Bok Lee, Dan Pei, MohammadTaghi Hajiaghayi, Ioannis Pefkianakis, Songwu Lu, He Yan, Zihui Ge, Jennifer Yates, and Mario Kosseifi. 2012. Threshold compression for 3g scalable monitoring. In *INFOCOM, 2012 Proceedings IEEE*. IEEE, 1350–1358.

[23] Dapeng Liu, Youjian Zhao, Haowen Xu, Yongqian Sun, Dan Pei, Jiao Luo, Xiaowei Jing, and Mei Feng. 2015. Opprentice: Towards Practical and Automatic Anomaly Detection Through Machine Learning. In *Proceedings of the 2015 ACM Conference on Internet Measurement Conference (IMC '15)*. ACM, New York, NY, USA, 211–224. https://doi.org/10.1145/2815675.2815679

[24] Wei Lu and Ali A Ghorbani. 2009. Network anomaly detection based on wavelet analysis. *EURASIP Journal on Advances in Signal Processing* 2009 (2009), 4.

[25] Ajay Mahimkar, Zihui Ge, Jia Wang, Jennifer Yates, Yin Zhang, Joanne Emmons, Brian Huntley, and Mark Stockert. 2011. Rapid detection of maintenance induced changes in service performance. In *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*. ACM, 13.

[26] Gerhard Münz, Sa Li, and Georg Carle. 2007. Traffic anomaly detection using k-means clustering. In *GI/ITG Workshop MMBnet*.

[27] Miguel Nicolau, James McDermott, et al. 2016. One-Class Classification for Anomaly Detection with Kernel Density Estimation and Genetic Programming. In *European Conference on Genetic Programming*. Springer, 3–18.

[28] Thomas Dyhre Nielsen and Finn Verner Jensen. 2009. *Bayesian networks and decision graphs*. Springer Science & Business Media.

[29] Brandon Pincombe. 2005. Anomaly detection in time series of graphs using arma processes. *Asor Bulletin* 24, 4 (2005), 2.

[30] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32 (ICML'14)*. JMLR.org, Beijing, China, II–1278–II–1286.

[31] Terrence J Sejnowski and Charles R Rosenberg. 1987. Parallel networks that learn to pronounce English text. *Complex systems* 1, 1 (1987), 145–168.

[32] Shashank Shanbhag and Tilman Wolf. 2009. Accurate anomaly detection through parallelism. *Network, IEEE* 23, 1 (2009), 22–28.

[33] Maximilian Sölch, Justin Bayer, Marvin Ludersdorfer, and Patrick van der Smagt. 2016. Variational inference for on-line anomaly detection in high-dimensional time series. *International Conference on Machine Laerning Anomaly detection Workshop* (2016).

[34] Jonathan AC Sterne, Ian R White, John B Carlin, Michael Spratt, Patrick Royston, Michael G Kenward, Angela M Wood, and James R Carpenter. 2009. Multiple imputation for missing data in epidemiological and clinical research: potential and pitfalls. *Bmj* 338 (2009), b2393.

[35] Hao Wang and Dit-Yan Yeung. 2016. Towards Bayesian deep learning: A survey. *arXiv preprint arXiv:1604.01662* (2016).

[36] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng, J. Chen, Z. Wang, and H. Qiao. 2018. Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications. *ArXiv e-prints* (Feb. 2018). arXiv:cs.LG/1802.03903

[37] Asrul H Yaacob, Ian KT Tan, Su Fong Chien, and Hon Khi Tan. 2010. Arima based network anomaly detection. In *Communication Software and Networks, 2010. ICCSN'10. Second International Conference on*. IEEE, 205–209.

[38] He Yan, Ashley Flavel, Zihui Ge, Alexandre Gerber, Dan Massey, Christos Papadopoulos, Hiren Shah, and Jennifer Yates. 2012. Argus: End-to-end service anomaly detection and localization from an ISP's point of view. In *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2756–2760.