

Robust and Unsupervised KPI Anomaly Detection Based on Conditional Variational Autoencoder

Zeyan Li^{*†}, Wenxiao Chen^{*†}, Dan Pei^{*†‡}

^{*}Tsinghua University

[†]Beijing National Research Center for Information Science and Technology(BNRist)

Email: {zy-li14, chen-wx17}@mails.tsinghua.edu.cn, peidan@tsinghua.edu.cn

Abstract—To ensure undisrupted web-based services, operators need to closely monitor various KPIs (Key Performance Indicator, such as CPU usages, network throughput, page views, number of online users, and etc), detect anomalies in them, and trigger timely troubleshooting or mitigation. There can be hundreds of thousands to even millions of KPIs to be monitored, thus operators need *automatic* anomaly detection approaches. However, neither traditional statistical approaches nor supervised ensemble approaches satisfy this requirement in practice when facing large number of KPIs. A state-of-art unsupervised approach *Donut* offering promising results, but it is not a sequential model thus cannot deal with the time information related anomalies. Thus, in this paper we propose *Bagel*, a robust and unsupervised anomaly detection algorithm for KPI that can handle time information related anomalies, using CVAE to incorporate time information and dropout layer to avoid overfitting. Our experiments using real data from Internet companies show that, compared to *Donut*, *Bagel* improves the anomaly detection best F1-score by 0.08 to 0.43.

I. INTRODUCTION

To ensure undisrupted web-based services, operators need to closely monitor various KPIs (Key Performance Indicator, such as CPU usages, network throughput, page views, number of online users, and etc), detect anomalies in them, and trigger timely troubleshooting or mitigation. Fig. 1 shows a few KPIs that we studied in this paper. There can be hundreds of thousands to even millions of KPIs to be monitored [1], [2], thus operators need *automatic* anomaly detection approaches.

Despite many proposed anomaly detection approaches in the past [3]–[12], most do not work well in the practice according to [11]. A comparison summary is shown in Table I, which will be elaborated later in §VI. For *Traditional statistical algorithms* [3]–[7], operators have to manually select an anomaly detection algorithm and tune its parameters for each KPI. *Supervised learning* based methods [8], [9] require manually labeling anomalies for each KPI. Thus, neither traditional statistical approaches nor supervised learning based approaches are automatic, and they do not work well in practice when facing large number of KPIs.

More recently, unsupervised approaches using deep generative models show some very promising results. Based on variational autoencoder (VAE), a state-of-art unsupervised anomaly detection algorithm, *Donut* [11], significantly outperforms

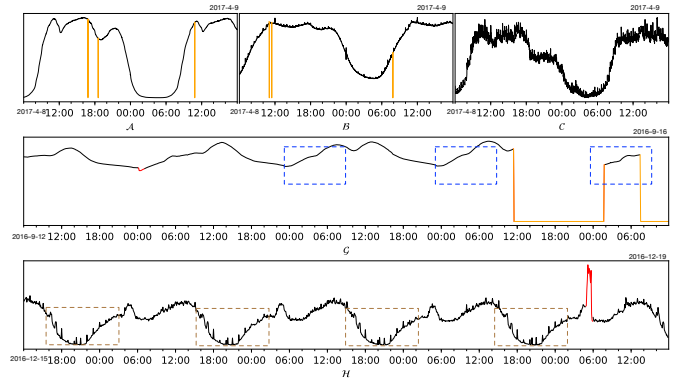


Fig. 1. KPI studied in this paper. We plot 36 hours of $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{H}$ and 72 hours of \mathcal{G} . The red lines denotes anomaly parts, and the orange lines denotes missing parts. The rest black lines are normal parts. $\mathcal{A}, \mathcal{B}, \mathcal{C}$ are very similar to those KPIs on which *Donut* works well. \mathcal{G} has a lot of missing points, and many long missing fragments. There are several normal fragments surrounded by long missing fragments. For example, the fragments in the blues boxes are following the same pattern, but the last one is surrounded by two long missing fragments, making it difficult to reconstruct its normal pattern. \mathcal{H} is quite smooth but has many short periodic spikes, but these spikes are not exactly the same every day (such as the valleys highlighted by the brown boxes).

the state-of-art supervised ensemble approach Opprentence (which outperforms all traditional statistical approaches) [9] on seasonal KPIs (such as $\mathcal{A}, \mathcal{B}, \mathcal{C}$ in Fig. 1). Seasonal KPIs are very common in practice and are business-related (such as number of online users, number of queries), thus are very important in anomaly detection [11].

However, *Donut* is not robust enough against time information related anomalies. This is because VAE is not a sequential model and *Donut* uses sliding windows to feed KPIs to VAE but ignores the relationship between windows. *i.e.*, *Donut* ignores the time information of a KPI window, and the SGD based optimization algorithm *Donut* uses shuffles the training data. For example, the relatively long fragment of missing data in \mathcal{G} in Fig. 1 causes false positives for the data points right after the missing data fragment (highlighted in rightmost blue dashed box in the figure), while we can imagine if we somehow incorporate timing information into the model (the pattern in those blue boxes are very similar across different days), these false negatives can be avoided. Similarly, the periodic (thus normal) spikes at the daily valleys in KPI \mathcal{H} in Fig. 1 will be mistakenly classified as anomalous by *Donut*, while incorporating timing information can help as well.

Since *Donut* is the state-of-art anomaly detection algorithm

Dan Pei[‡] is the corresponding author.

978-1-5386-6808-5/18/\$31.00 ©2018 IEEE

TABLE I
COMPARISON AMONG ANOMALY DETECTION METHODOLOGIES

Suffers from	1	2	3	4	5	<i>Bagel</i>
Selecting algorithm	Yes	No	Some	No	No	No
Tuning parameters	Yes	No	Some	Some	Some	No
Relying on labels	No	Yes	No	No	No	No
Poor Capacity	Yes	No	Some	No	No	No
Hard to train	No	No	Some	Some	Yes	No
Time consuming	Some	Yes	Some	No	No	No

1: traditional statistical method, e.g., time series decomposition [5]

2: supervised ensemble method, e.g., Opprentice [9]

3: traditional unsupervised method, e.g., one-class SVM [14]

4: sequential deep generative model, e.g., VRNN [15]

5: non-sequential deep generative model, e.g. VAE [11], [16]

and more importantly have solid theoretical foundations, in this paper, we aim to push *Donut* one significant step forward towards practical deployment, by improving *Donut*'s robustness against time information related anomalies, like \mathcal{G}, \mathcal{H} in Fig. 1. Note that in this paper we focus on anomaly detection in seasonal KPIs as well, just as in *Donut* [11].

We propose *Bagel*, a robust and unsupervised anomaly detection algorithm for KPI. To incorporate time information, *Bagel* is based on conditional variational autoencoder (CVAE) as opposed to VAE in *Donut*, and uses time information as the input condition. However, there is one important challenge in incorporating time information into CVAE model. Because fitting the relationship between timing and KPI value is much easier (similar to traditional statistical model with seasonality, e.g., historical average) than fitting the relationship between the input sliding windows and reconstructed normal patterns, CVAE can be easily overfitted on time information for seasonal KPIs. To avoid overfitting, we add an extra layer of dropout, which can be considered making an ensemble model of many smaller neural networks [13].

The contributions of this paper can be summarized as follows:

- For the first time in the literature, we identify the importance of time information for non-sequential deep generative models, such as *Donut*, in KPI anomaly detection problem.
- To the best of our knowledge, *Bagel* is the first to apply conditional variational autoencoder (CVAE) to KPI anomaly detection and use dropout technique to successfully avoid overfitting.
- Our experiments using real data from Internet companies show that, compared to *Donut*, *Bagel* improves the anomaly detection best F1-score by 0.08 to 0.43 for KPIs \mathcal{G} and \mathcal{H} , greatly improving *Donut*'s robustness against time information related anomalies.

The rest of the paper is organized as follows. §IV-A reviews the background of KPI anomaly detection and reviews the background of VAE and CVAE. §III presents *Bagel*'s neural network architecture, and its design of training and anomaly detection. §IV evaluates *Bagel*'s performance. §V analyzes how *Bagel* works. §VI reviews related work. Finally, §VII concludes the paper.

II. BACKGROUND AND PROBLEM

A. KPI and KPI Anomaly Detection

In this paper, we focus on business-related KPIs, as in [11]. These KPIs have *seasonal* patterns because of the influence from user behavior and schedule. However, the KPI patterns at each repetitive cycle is *not* exactly the same, since user behavior will not be exactly the same everyday. As [11] does, we name these differences “local variations”. A KPI anomaly detection algorithm will not work well unless it can handle the local variations well. Besides the seasonal pattern and local variations, there are also noises on KPIs. We assume that the noises follow independent, zero-mean Gaussian distribution.

In summary, the *normal patterns* of the KPIs that we study consist of two components: (1) seasonal patterns with local variations, (2) independent, zero-mean Gaussian noises. The anomalies are those data points which do not follow the normal patterns.

The KPI values are usually collected with a fixed monitoring interval like 10 seconds or 1 minute. However, because of occasional technical errors, sometimes the KPI values are not collected. These data points are called missing points. Missing points are also some kind of anomalies, but it is easy to distinguish them from normal points. Therefore, in this paper, we use *anomaly points* to call those points that do not follow normal patterns but are not missing points, and use *abnormal points* to call both missing points and anomaly points.

KPI anomaly detection problem can be formulated as follows: for any time t , given historical KPI observations $v_{t-W+1:t}$ with length W , determine whether anomaly happens at time t (denoted by $\gamma_t = 1$).

B. Variational Autoencoder and Conditional Variational Autoencoder

As previously mentioned in §I, one particularly promising direction for KPI anomaly detection is deep generative model, such as Variational Auto-Encoder (VAE) [17], [18]. VAE uses neural networks to model data's distribution and generate new samples following it. *Donut* [11] is a state-of-art VAE based KPI anomaly detection algorithm. In this section, we briefly introduce the background of VAE and conditional variational auto-encoder (CVAE) used in our proposed approach *Bagel*.

Deep Bayesian networks combines deep learning and probabilistic graphical models (PGM). It models the relationship among random variables with neural networks, which extends the ability of PGM. Variational inference [19] is very useful for solving the posteriors of the distributions derived by neural networks, so it is usually adopted for the training and prediction of deep Bayesian networks.

Variational autoencoder (VAE) and conditional variational autoencoder (CVAE) [20], [21] are typical deep Bayesian networks. VAE models the relationship between two random variables \mathbf{x} and \mathbf{z} . CVAE models the relationship between \mathbf{x} and \mathbf{z} , conditioned on \mathbf{y} , i.e., it models $p(\mathbf{x}, \mathbf{z}|\mathbf{y})$. VAE and CVAE are very similar. We choose CVAE in *Bagel* rather

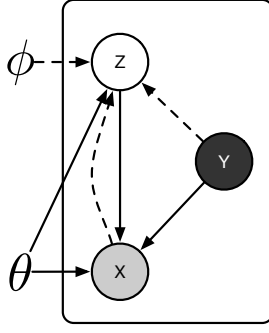


Fig. 2. The architecture of CVAE. Considering the prior of \mathbf{z} as part of the generative process, the whole generative model (solid lines) can be formulated as $p_\theta(\mathbf{z}, \mathbf{x}|\mathbf{y}) = p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{y})p_\theta(\mathbf{z}|\mathbf{y})$. The approximated posterior (dashed lines) is $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})$.

than VAE because the condition variable is important for KPI anomaly detection (see §V-B).

The generative process of CVAE is as follows:

- 1) Choose a \mathbf{z} prior distribution, and sample \mathbf{z} from it, i.e., $\mathbf{z} \sim p_\theta(\mathbf{z}|\mathbf{y})$. As [21] suggests, we can make latent \mathbf{z} independent of \mathbf{y} , i.e., $\mathbf{z} \sim p_\theta(\mathbf{z})$.
- 2) Sample \mathbf{x} from $p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{y})$, which is derived from a neural network with parameter θ , i.e., $\mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{y})$

Although the true posterior $p_\theta(\mathbf{z}|\mathbf{x}, \mathbf{y})$ plays an important role in training and prediction, it is intractable [18]. In variational inference, it is approximated by a variational distribution $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})$, which is fitted by another neural network with parameter ϕ . SGVB [17], [18] is a variational inference algorithm which is often used along with VAE. SGVB jointly trains the approximated posterior and generative model by maximizing the evidence lower bound (ELBO, Eqn. (1)). We adopt SGVB because it works for a broad range of applications [19] and is sufficient for our task already.

$$\begin{aligned} & \log p(\mathbf{x}|\mathbf{y}) \\ & \geq \log p(\mathbf{x}|\mathbf{y}) - \text{KL}[q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y}) \| p_\theta(\mathbf{z}|\mathbf{x}, \mathbf{y})] \\ & = \mathcal{L}(\mathbf{x}, \mathbf{y}) \\ & = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})} [\log p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{y}) + \log p_\theta(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})] \end{aligned} \quad (1)$$

The overall architecture of CVAE is summarized in Fig. 2

III. ARCHITECTURE

In this section, we will introduce the details of our proposed algorithm *Bagel*, including the network architecture, training and detection. We will also highlight the major difference between *Bagel* and *Donut*.

A. Network Architecture

1) *Preprocessing*: As mentioned in §II-A, there are some missing points in KPIs. So first of all we impute these missing points with zero, then the imputed KPIs become time series with fixed monitoring interval. Different KPIs' value ranges are various, so we standardize KPIs with z-score: firstly we calculate the mean μ and standard deviation σ of the whole KPI, and then calculate the new value of each point v_i by $v_i \leftarrow \frac{v_i - \mu}{\sigma}$.

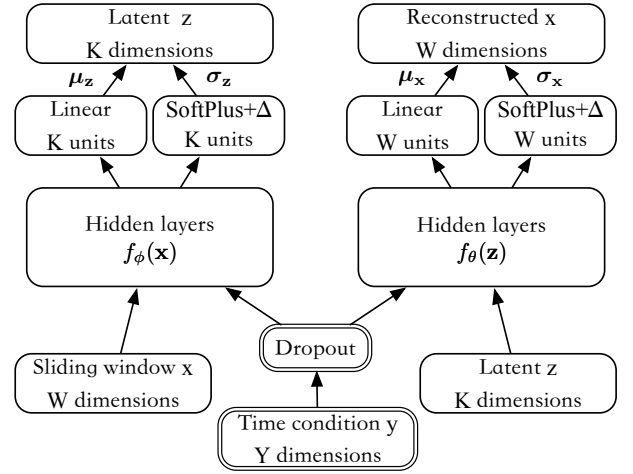


Fig. 3. The overall neural network architecture. The double-lines highlight the major difference with *Donut* in network architecture.

As mentioned in §II-A, the KPIs that we studied are time series. However, CVAE is not a sequential model. We use sliding windows of a KPI as the input data of CVAE. Formally speaking, for a KPI $\mathbf{v} = (v_1, v_2, \dots, v_n)$, the i -th window of the KPI is $\mathbf{x}^{(i)} = (v_i, v_{i+1}, \dots, v_{i+W-1})$, where W denotes the window's length.

2) *Architecture*: The overall neural network architecture is shown in Fig. 3. As [21] suggests, we make the latent variable \mathbf{z} independent of condition variable \mathbf{y} . The \mathbf{z} prior is chosen to be $p(\mathbf{z}|\mathbf{y}) = p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. Both \mathbf{z} and \mathbf{x} posterior are chosen to be diagonal Gaussian distributions, i.e., $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y}) = \mathcal{N}(\mu_z, \text{diag}(\sigma_z^2))$, $p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{y}) = \mathcal{N}(\mu_x, \text{diag}(\sigma_x^2))$, where $\mu_z, \mu_x, \sigma_z, \sigma_x$ denote the means and standard deviations of $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})$ and $p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{y})$. It makes sense since we already assume that there are independent Gaussian noises on the KPIs. The hidden neural networks $f_\phi(\mathbf{x})$ and $f_\theta(\mathbf{z})$ are both several fully connected layers with ReLU [19] activation. They are used to extract hidden features from \mathbf{z} or \mathbf{x} for deriving Gaussian statistics. The Gaussian means are derived with a linear layer: $\mu_z = \mathbf{W}_z^\top f_\phi(\mathbf{x}) + \mathbf{b}_{\mu_z}$, $\mu_x = \mathbf{W}_x^\top f_\theta(\mathbf{z}) + \mathbf{b}_{\mu_x}$. The standard deviations are derived by a softplus layer plus a positive real constant Δ : $\sigma_z = \ln(1 + \exp(f_\phi(\mathbf{x}))) + \Delta$, $\sigma_x = \ln(1 + \exp(f_\theta(\mathbf{z}))) + \Delta$. Softplus gives very similar outputs with the common activation function ReLU [19], but the outputs are all strictly positive, which is required by standard deviation. The positive real constant Δ is used to avoid numeric problems, such as underflow.

3) *Encoding Time Information*: The condition variable \mathbf{y} represents the input window \mathbf{x} 's timestamp (to be precise, the latest point \mathbf{x}_W 's timestamp)

To emphasize the seasonality, the timestamp are decomposed into several parts: minute, hour, day of week, since user behavior schedule usually can be factorized to these basic units. We do not use seconds because the interval of KPIs is typically 1 minute or 5 minutes in our context. We do not use month or year since empirically there is no seasonality at these levels. Since neural networks is more sensitive to directions

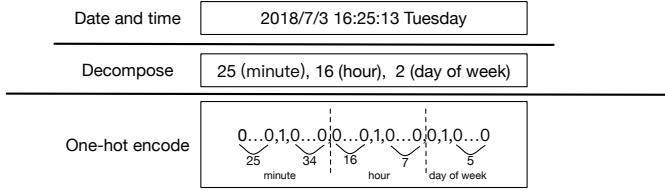


Fig. 4. The way we construct condition variable for window \mathbf{x} . The first line is the time of a window (specifically speaking, the timestamp of the latest point of the window). Then we use some components of the timestamp as the input condition for CVAE: the minute 25, the hour 16, the day of week: 2 (for Tuesday). We do not use second or month and year, because typically there is no seasonality of these levels in our data. Then we convert these three numbers into three one-hot encoded vectors and concatenate them. One-hot encoding means converting a positive integer to a binary vector full of zeros but only a single one, and the position of the single one represents the original integer's value. For example, a "1" value the 26th element in the *minute* vector means *minute* = 25 in the timestamp.

than values [19], we choose to convert the decomposed values to one-hot vectors. Fig. 4 illustrates how *Bagel* encodes time information.

Encoding time information can help *Bagel* deal with time information related anomalies. For example, in \mathcal{G} , too many missing points makes it hard to reconstruct normal patterns from the sliding windows, but, with the help of time information, *Bagel* is less affected. In \mathcal{H} , the spikes are not exactly the same every day, so the reconstruction may be biased. Since H are otherwise quite smooth except for those periodic spikes, *i.e.*, the \mathbf{x} standard deviation is fairly small, those little biases cause unreasonable high anomaly scores from *Donut*. With the help of time information, *Bagel* is not confused by these spikes.

Because fitting the relationship between timing and KPI value are easier (similar to traditional statistical model with seasonality, *e.g.*, historical average) than fitting the relationship between \mathbf{x} and reconstructed normal patterns, CVAE can be easily overfitted on time information for KPI anomaly detection. As will be demonstrated in §V-B, the performance of *Bagel* without dropout layer is really poor. This is, without dropout layer, the model pays too much attention to the relationship between time and the normal patterns, *i.e.*, it learns too much about the seasonality but too little about the local variations. However, as [11] points out, an anomaly detection algorithm cannot work well unless local variations are handled appropriately.

Therefore, in *Bagel*, there is an input dropout layer for condition variable \mathbf{y} , as shown in Fig. 3. Dropout [13] reduces the risk of overfitting by randomly disabling some connections in a neural network in training. A network with dropout layers can be considered as an ensemble of many smaller networks [13]. Since we notice that CVAE can be easily overfitted on \mathbf{y} , we add an extra dropout layer after the condition input layer. The input dropout in *Bagel* is implemented by randomly setting p_{dropout} ratio of dimensions to zero. In the perspective of ensemble as [13], now *Bagel* becomes an ensemble model of many smaller models, which only take a small part of \mathbf{y} as the input condition variable. Fitting the relationship between

time information and normal patterns are considered easy, but fitting that between only a small part of time information and normal patterns will not be so easy, thus we avoid overfitting.

4) *Summary of improvement over Donut*: Compared to the network architecture in *Donut*, *Bagel*'s major improvement are three-fold:

- Adoption of CVAE as opposed to VAE so that we can encode time information as the conditional variable.
- Use one-hot encoding to encode time information in vectors.
- Use dropout layer to avoid overfitting introduced by time information.

B. Training

The SGVB algorithm [17], [18] is used for training CVAE. One key technique for SGVB is re-parameterization, which means in training \mathbf{z} is produced by $\mathbf{z} = \epsilon_{\mathbf{z}} \cdot \sigma_{\mathbf{z}} + \mu_{\mathbf{z}}$, $\epsilon_{\mathbf{z}} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$, rather than $\mathbf{z} \sim \mathcal{N}(\mu_{\mathbf{z}}, \text{diag}(\sigma_{\mathbf{z}}^2))$. It makes it possible to pass gradients to the approximated posterior $q_{\phi}(\mathbf{z}|\mathbf{x}, \mathbf{y})$ and thus train it.

The CVAE model is supposed to capture the normal patterns of a KPI, thus it is necessary to avoid learning the abnormal patterns. [11] proposes a loss function called M-ELBO to ignore abnormal pattern for VAE and shows that M-ELBO is satisfying enough. A similar loss function can be applied to CVAE. For an input window \mathbf{x} , assume that its corresponding label window is a binary vector α (\mathbf{x}_i is abnormal if and only if $\alpha_i = 1$), and β denotes the proportion of normal points. Since the \mathbf{x} posterior is assumed to be diagonal Gaussian distribution, its log-likelihood can be rewritten as $\log p_{\theta}(\mathbf{x}|\mathbf{z}, \mathbf{y}) = \sum_{i=1}^W \log p_{\theta}(\mathbf{x}_i|\mathbf{z}, \mathbf{y})$. By multiplying α_i and $\log p_{\theta}(\mathbf{x}_i|\mathbf{z}, \mathbf{y})$, the M-ELBO for CVAE can be formulated as follows:

$$\tilde{\mathcal{L}}(\mathbf{x}, \mathbf{y}) = \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x}, \mathbf{y})} \left[\sum_{i=1}^W \alpha_i \cdot \log p(\mathbf{x}_i|\mathbf{z}, \mathbf{y}) + \beta \cdot \log p(\mathbf{z}|\mathbf{y}) - \log q_{\phi}(\mathbf{z}|\mathbf{x}, \mathbf{y}) \right] \quad (2)$$

M-ELBO makes the CVAE model ignore the loss from abnormal points in a training window \mathbf{x} . Though there may be only occasional labels, ignoring missing points can be very helpful [11]. Therefore it makes our model able to learn reconstructing normal patterns from potential abnormal windows. To take advantage of such an ability, abnormal data injection, which is some kind of data augmentation should be applied. However, since CVAE itself is a generative model with high capacity, using another simpler generative model to generate anomaly points seems unreasonable. Therefore only missing points are injected in our practice. Before each epoch, some points are randomly chosen and set to zero (or other imputed value for missing points).

Bagel's training design is similar to that of *Donut*, and the difference is that we need to introduce conditional variable \mathbf{y} in probability functions in *Bagel*. We do not claim the *Bagel*'s training design as our contribution.

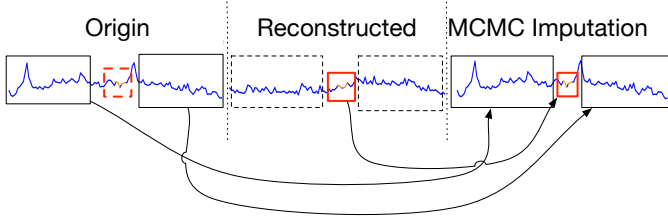


Fig. 5. Illustration of a MCMC imputation step. Firstly, use the trained CVAE model to reconstruct the original input vector \mathbf{x} , then replace the missing points with those corresponding points in the reconstructed vector.

C. Detection

We use the reconstruction term in ELBO (Eqn. (1)) as anomaly detector, *i.e.*, $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})} [\log p_\theta(\mathbf{x}|\mathbf{z},\mathbf{y})]$. It is called “reconstruction probability” in [11], [16], while it is actually not a well-defined probability. As mentioned above, the CVAE model are supposed to learn reconstructing normal patterns from potential abnormal input window \mathbf{x} . If a sample of \mathbf{z} from $q_\phi(\mathbf{z}|\mathbf{y},\mathbf{x})$, $\mathbf{z}^{(i)}$, is corresponding to \mathbf{x} ’s normal patterns (*i.e.*, $p_\theta(\tilde{\mathbf{x}}|\mathbf{z}^{(i)},\mathbf{y})$ gives \mathbf{x} ’s normal patterns’ distribution), then the log-likelihood $\log p_\theta(\mathbf{x}|\mathbf{z}^{(i)},\mathbf{y})$ represents how much \mathbf{x} follows the normal patterns. Considering $\mathbf{z}^{(i)}$ ’s likelihood $q_\phi(\mathbf{z}^{(i)}|\mathbf{x},\mathbf{y})$ as a weight, the negative weighted average $-\sum_{\mathbf{z}^{(i)}} q_\phi(\mathbf{z}^{(i)}|\mathbf{x},\mathbf{y}) \log p_\theta(\mathbf{x}|\mathbf{z}^{(i)},\mathbf{y})$ becomes a reasonable anomaly score. Its expectation form is $-\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})} [\log p_\theta(\mathbf{x}|\mathbf{z},\mathbf{y})]$.

The abnormal points in the testing windows may have a bad influence on finding a good posterior. In order to reduce the bias introduced by missing points (we don’t know which points are anomaly points), we take advantage of the trained generative CVAE model to impute the missing points in the testing windows with MCMC [17]. At each step, we feed a testing window \mathbf{x} to the trained model, replace the missing points of \mathbf{x} with the corresponding missing points in the reconstructed window, and keep the other points original. Such a step is repeated for L times. An illustration of MCMC imputation is given in Fig. 5

Bagel’s design of detection is similar to that of *Donut*, and the difference is that we need to introduce conditional variable \mathbf{y} in probability functions in *Bagel*. We do not claim the *Bagel*’s detection design as our contribution.

D. Improvement over Donut

The main differences from *Donut* are highlighted in Fig. 3 with double lines. Firstly, we adopt CVAE rather than VAE, so there is an input condition in both variational and generative networks. We also use sliding windows as *Donut* does, but with the help of the additional time condition, *Bagel* will not ignore the time information of the KPI windows. Secondly, to reduce the risk of overfitting, we add an additional dropout layer after the condition input layer.

These design differences make *Bagel* more robust than *Donut* when dealing with time information related anomalies. Later in §IV-A will show that *Bagel* outperforms *Donut* on

KPIs such as KPIs \mathcal{H} and \mathcal{G} which have time information related anomalies, and §V will explain the results in detail.

Since the condition variable actually do not affect the overall architecture a lot, *Bagel* re-uses some designs in *Donut*: we use the same preprocessing methods, and use M-ELBO and reconstruction probability with condition as *Bagel*’s training objective and detector. We also adopt the missing data injection and MCMC imputation techniques from *Donut* as they are shown to be effective in [11]. We do not claim these are the contributions of *Bagel*.

IV. EXPERIMENTS

In this section, we use real data from Internet companies to evaluate *Bagel*’s performance and compare with some state-of-art algorithms.

A. Evaluation Metrics

A modified anomaly F1-score, which is proposed by [11], will be used as our evaluation metric. F1-score is usually adopted to evaluate classification problems, which takes both precision and recall into consideration. However, in KPI anomaly detection problem, the points to be classified are not independent and operators only cares about the times when anomalies start [11]. Therefore, if an anomaly detection algorithm raises an alert fast enough (*i.e.*, before a maximum allowed delay) after an anomaly begins, the whole anomaly fragment will be considered detected successfully when calculating modified F1-score. A successful alert with huge delay is not useful at all. Thus, if for an anomaly segment in ground truth, an anomaly detection algorithm does not raise any alert before the maximum allowed delay, this whole anomaly segment in the ground truth is considered as false negative, even though the anomaly detection algorithm might raise some alerts after the maximum allowed delay.

An illustration of the metric is given in Fig. 6. For convenience, we will call this modified F1-score as just F1-score. Similar to [11], to show the best potential performance of the models, we use the best F1-score, which is computed with the threshold which acquires best performance on **test** set. In other words, we calculate the modified F1-scores on test set with all potential thresholds and use the best one as our evaluation metric.

B. Datasets

We obtain several well-maintained KPIs from several large Internet companies. All the anomaly labels are manually confirmed by operators. The statistics of these KPIs are shown in Table II. $\mathcal{A}, \mathcal{B}, \mathcal{C}$ are similar to those in [11], so they can demonstrate *Bagel*’s performance on those KPIs that *Donut* claims to handle well. \mathcal{G} has many missing points and several long missing fragments (like that shown in Fig. 1, and there are several similar long missing fragments), such that many normal fragments are just small pieces surrounded by missing points. \mathcal{H} is quite smooth, but has many periodic spikes every day. \mathcal{G} and \mathcal{H} are representative KPIs to demonstrate the effect of time information. Since we collect these datasets from

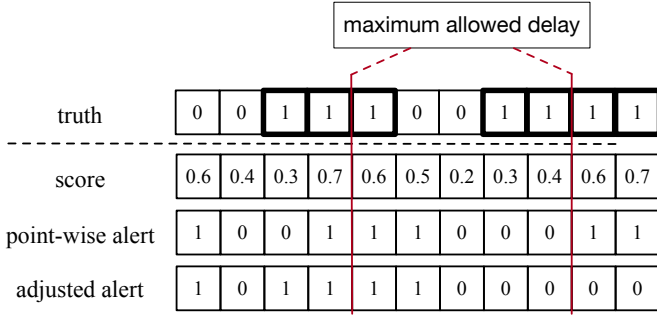


Fig. 6. Illustration of the evaluation metric. The first line is the ground truth and the second line is the anomaly scores. With the threshold 0.5, the original point-wise alerts are given in the third line. The maximum allowed delay is set to be 1 here (the vertical lines). In the first anomaly segment, there is an alert before the maximum allowed delay, so the whole fragment is considered to be detected successfully. In the second anomaly segment, the first alert occurs after the maximum allowed delay, so the whole fragment is not considered to be detected at all.

TABLE II
STATISTICS OF KPI DATASETS

KPI	total points	missing points	anomaly points	anomaly fragments	monitoring interval
\mathcal{A}	296460	1222/0.412%	1213/0.409%	51	1min
\mathcal{B}	317522	1117/0.352%	1979/0.623%	49	1min
\mathcal{C}	285120	304/0.107%	4394/1.541%	126	1min
\mathcal{G}	24950	3032/12.152%	365/1.463%	31	5min
\mathcal{H}	17568	0/0.000%	103/0.586%	6	5min

different sources, they have different monitoring intervals and different number of data points. Fig. 1 plots these KPIs.

C. Overall Performance on $\mathcal{A}, \mathcal{B}, \mathcal{C}$

We compare *Bagel*'s performance with that of *Donut* (which is the state-of-art unsupervised algorithm) and *Opprentice* (which is the state-of-art supervised algorithm and outperforms most traditional statistical algorithms [9]). All experiments are repeated for 10 times.

In the following experiments, we set latent space dimensions $K = 8$, missing data injection radio $\lambda = 0.01$, dropout rate $p_{\text{dropout}} = 0.1$. We use two fully-connected layers with 100 units in both variational and generative networks. *Donut* uses the same number of layers with the same number of units. *Opprentice* uses random forest (the number of components is 200 and the max depth is 6) to ensemble 129 traditional detectors. We set the maximum allowed delay in modified F1-score to be 7 as in [11]. *Bagel* and *Donut* are both trained without any labels, and *Opprentice* need complete anomaly labels.

Average best F1-scores over different W are shown in Fig. 7. On datasets $\mathcal{A}, \mathcal{B}, \mathcal{C}$, *Bagel*'s performance is similar to that of *Donut*'s, which means *Bagel* is also able to handle those KPIs that *Donut* is able to handle.

D. Overall Performance on \mathcal{G}, \mathcal{H}

In Fig. 8, we compare the performance of *Bagel*, *Donut* and *Opprentice* on KPI \mathcal{G}, \mathcal{H} .



Fig. 7. Best F1-score with different window sizes on $\mathcal{A}, \mathcal{B}, \mathcal{C}$. *Bagel* and *Donut* have very similar performance. The average F1-score of *Bagel* ranges from 0.7 to 0.8 on $\mathcal{A}, \mathcal{B}, \mathcal{C}$ in most settings.

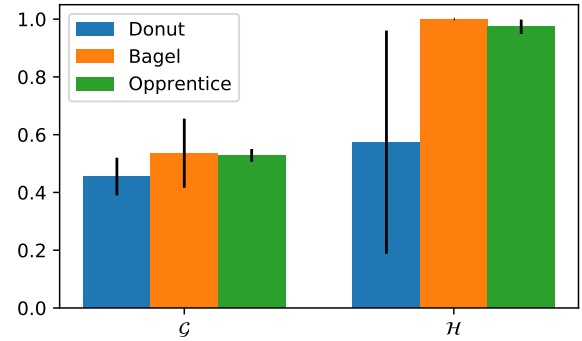


Fig. 8. Best F1-score on \mathcal{G}, \mathcal{H} . On \mathcal{G} , *Bagel* outperforms *Donut* by 0.08. On \mathcal{H} , *Bagel* outperforms *Donut* by 0.43. *Opprentice* always get similar performance with *Bagel* on these two KPIs, but it is supervised algorithm while the others are unsupervised.

For \mathcal{G} , *Bagel*'s average best F1-score is 0.08 higher than that of *Donut*'s, and *Opprentice* has similar performance with *Bagel*. A comparison of the anomaly scores of \mathcal{G} given by *Donut* and *Bagel* are given in Fig. 9. The small normal pieces surrounded by missing fragments (such as that shows in Fig. 9) is hard to reconstruct for *Donut*, because too many points are missing and *Donut* does not have enough information to reconstruct the normal pattern. Therefore *Donut* may give too high anomaly scores for a normal window, which will cause poor performance. With the help of the additional time information, *Bagel* is less affected by the large amount of missing points, since this KPI have very similar patterns at the same time every day and *Bagel* is supposed to find this fragment's normal pattern in these patterns. *Opprentice* also outperforms *Donut* on \mathcal{G} , because *Opprentice* have many detectors that will not be affected by a large amount of missing points (e.g., the difference from the KPI value from the last season).

Bagel's average best F1-score on \mathcal{H} is 0.43 more than that of *Donut*'s. Fig. 10 shows that anomaly scores of \mathcal{H} given by *Donut* and *Bagel*. Since \mathcal{H} is very smooth at most points, the x 's standard deviation will be quite small (nearly zero).

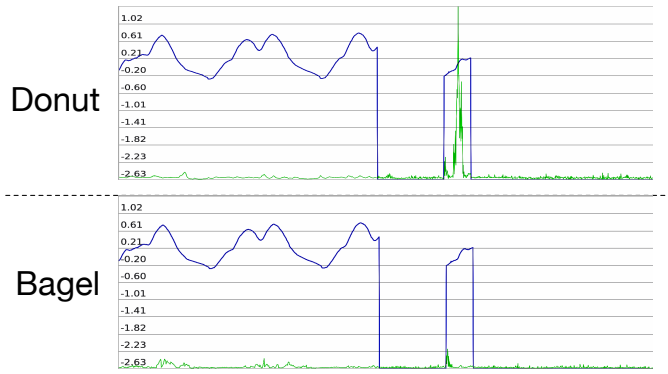


Fig. 9. Anomaly scores of \mathcal{G} given by *Donut* and *Bagel*. The blue lines are KPI values and the red line marks the ground truth anomalies. The green lines are the anomaly scores for each point. *Donut* gives too high anomaly scores for the normal fragment surrounded by missing points.

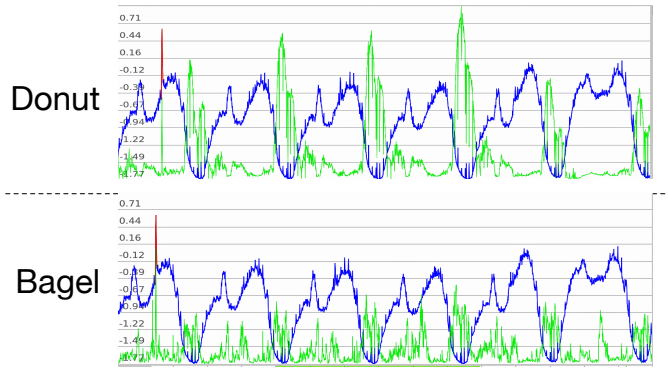


Fig. 10. Anomaly scores of \mathcal{H} given by *Donut* and *Bagel*. The blue lines are KPI values and the red line marks the ground truth anomalies. The green lines are the anomaly scores for each point. *Donut* gives too high anomaly scores at many normal valleys, which are mostly smooth but have many periodic spikes.

However, since the periodic spikes do not occur at exactly the same time every day, and do not have the same height every day, such spikes bring some extra bias for reconstruction. Small bias may also cause big impact on standard deviation since the standard deviation is too small on a mostly smooth KPI. Thanks to the time information, *Bagel* does not suffer so much from this. Although such spikes also bring extra bias for *Bagel*, the time information can reduce its impact by guiding the reconstruction. *Donut* finds normal patterns in all patterns that are similar to the input window, but *Bagel* will only find similar normal patterns in all patterns that have similar y . Note for a seasonal KPI, KPI windows with similar y are supposed to have similar patterns.

Donut's performance in \mathcal{H} is much worse than Opprentice because Opprentice has many detectors (thus features) and many of them are not affected by the small standard deviation. *Bagel* outperforms Opprentice a bit since CVAE has more capacity than simple statistical detectors.

V. ANALYSIS

In this section, we explain in detail how *Bagel*'s two important components work: conditional variable with time information, and dropout layer.

A. Conditional KDE explanation

The effectiveness of M-ELBO, missing data injection and MCMC imputation for VAE have already been shown in [11]. As the additional input condition in CVAE does not change the network structure a lot, these three techniques can be easily applied to CVAE and they are supposed to work in the same way as in VAE in [11]. Generally speaking, M-ELBO and the dimension reduction in autoencoder structure makes *Bagel* be able to reconstruct normal patterns from a potential abnormal window, and missing data injection amplifies M-ELBO's effect, and MCMC imputation help the model find a good posterior in detection. As a result, since *Bagel* also uses reconstruction probability as the detector, *Bagel* still works in the KDE way, as interpreted in [11].

The difference between *Bagel* and *Donut* is that *Bagel* finds z posterior conditioned on time information. In the KDE perspective, *Bagel* uses kernels and weights conditioned on time information. We call the way *Bagel* works as *conditional KDE*.

Given a KPI window, its corresponding normal patterns is multimodal. For example, suppose that a script should be executed every day at 3:00pm, so there will be a peak in the KPI at 3:00pm every day. If someday this script fails to be executed, the corresponding KPI fragment will look very similar to that of different time of day (e.g., 11:00am), where no scripts are executed. Without time information, we cannot tell if this fragment is normal (because it is similar to the normal pattern at 11:00am), or abnormal. However, with time information, we can confidently report that this fragment is abnormal since we know there is a peak at 3:00pm every day.

Time information also helps when x is confusing. For example, in Fig. 9, there is a normal fragment surrounded by missing points. As this normal fragment is much shorter than the training windows used to train the model, *Donut* cannot determine its normal pattern and, therefore, gives wrong anomaly scores. But *Bagel* have additional time information, so it is able to reconstruct this fragment's normal pattern even if too many points are missing. Although *Bagel* also cannot reconstruct its normal pattern by the small piece of normal points, the time information guides *Bagel* to find normal pattern in those patterns that occur at similar time in a day, which is much easier and robust than find normal patterns in all possible patterns.

B. Dropout for avoiding overfitting on time information

Modeling the relationship between latent variables and encoded timestamps is easier than that between latent variables and sliding windows, because the KPIs are mostly seasonal and the local variation is not so influential compared to the periodicity. Therefore CVAE model may be overfitted on time information easily. In order to avoid the risk of overfitting on timing, we use an extra dropout [13] layer after the condition variable's input layer (see Fig. 3).

The time gradient effect originally pointed out by [11] for *Donut*, in CVAE becomes " z samples drawn from approximated z posterior $q_\phi(z|x, y)$ with more different y should be

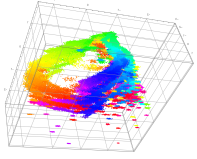
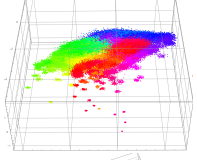
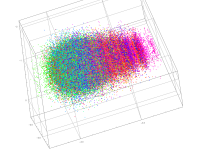
	Latent Space	Best F1-score
Bagel		0.686
Bagel without Dropout		0.605
Time Only		0.074

Fig. 11. Dropout’s effect on latent spaces. This figure plots the 3-D latent spaces of \mathcal{G} with different models. “*Bagel* without dropout” denotes our model without the dropout layer. “Time only” denotes a model use only time information in the encoder, *i.e.*, $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y}) = q_\phi(\mathbf{z}|\mathbf{y})$

far away from each other”. This time gradient is important to find a good \mathbf{z} posterior according to the analysis in [11]. If the \mathbf{z} samples of two \mathbf{x} windows following different patterns are not far away from each other but are fairly close, it will be much more likely to find wrong normal patterns.

Some latent spaces are shown in Fig. 11, which draws the \mathbf{z} samples: $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})$, $\mathbf{x}, \mathbf{y} \sim p_{\text{data}}(\mathbf{x}, \mathbf{y})$, and a point’s color denotes its corresponding time in a day (counting in minutes, *i.e.*, in $[0, 1440)$). We compare the latent spaces of our proposed model (with dropout layer), that without dropout layer, and a “time only” model, in which the shape information is ignored, *i.e.*, $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y}) = q_\phi(\mathbf{z}|\mathbf{y})$ and there is no dropout layer.

In the latent spaces of “time only” models, \mathbf{z} samples with different colors, *i.e.*, different times and therefore different shapes, are completely mixed together. As for *Bagel* without dropout, the points with similar colors also get together somewhat, but dissimilar points are not far away enough like those in *Bagel*’s latent space.

In the latent space of *Bagel*, the \mathbf{z} samples form a circle, and along the circle the colors changes gradually, which means that points with different \mathbf{y} are as far away as possible. Therefore, we conclude that dropout layer is helpful to learn a good \mathbf{z} space topology, and therefore helpful to reconstruct normal patterns.

Since the latent spaces of *Bagel* have significant time gradient, similar to that in *Donut* [11], *Bagel* has similar ability with *Donut* to reconstruct normal patterns from \mathbf{x} , but *Bagel* has timing information successfully incorporated without overfitting.

VI. RELATED WORK

This section briefly reviews three main directions in KPI anomaly detection: traditional statistical methods, supervised ensemble methods, and unsupervised learning based methods.

Many **traditional statistical methods** are used for anomaly detection in the past few decades, such as [1], [3]–[9], [14], [15], [22]–[33]. These methods make some simple assumptions for the KPIs, therefore operators must take efforts to choose an appropriate algorithm and tune the hyper-parameters for each KPI and each service. Simple equations used in these models are often not able to capture the properties of the KPIs in practice. For example, [5] detects the anomalies in search response time KPI with WoW (week-over-week) method, but there are not only weekly periodicity but also daily periodicity, holidays and other factors. In such cases the combination of many different detectors seems necessary. However, simple ensemble of these statistical detectors, like majority vote [24] or normalization [30], does not help a lot according to [9].

To automatically combine detectors, some **supervised ensemble methods**, such as EDGAS [8] and Opprentice [9], were proposed. The philosophy under them are clear and simple: since simple combination of detectors is not enough, we can use actual data to model the best combination of detectors. They compute features with traditional statistical detectors firstly, and then train a supervised machine learning classifier model like random forest, with user feedbacks, typically anomaly labels. These supervised methods show excellent performance. For example, experiments in [9] show that Opprentice outperforms all traditional statistical methods. However, they heavily rely on careful labels. However, getting enough careful labels requires too much manual efforts and time, which makes such supervised methods expensive and impractical. Although there might be occasional anomaly labels, the coverage of these labels is usually far from the requirements of traditional supervised methods. Besides, these ensemble algorithms are time-consuming because they have to execute a lot of traditional statistical algorithms, some of which might be time-consuming.

In general, anomaly detection (not necessarily KPI anomaly detection) based on **unsupervised machine learning**, such as one-class SVM [14], [23], GMM [26], VAE [11], [16], [34] and VRNN [15], model the **normal patterns** with machine learning methodology and raise alerts for points that do not follow normal patterns (rather than doing classification between normal and abnormal like Opprentice [9]). Along this direction, *Donut* [11] is so far the only one that works on KPI anomaly detection, and it outperforms the state-of-art supervised ensemble approach Opprentice. Our experiments have shown that *Bagel* outperforms it in robustness.

VII. CONCLUSION

Anomaly detection for the vast number of KPIs in the web-based services require automatic approaches. Built upon a state-of-art unsupervised VAE-based approach *Donut*, this paper for the first time identifies *Donut*’s inadequacy in dealing with the time information related anomalies, and proposes

Bagel, a robust and unsupervised KPI anomaly detection algorithm that can handle time information related anomalies. Compared to *Donut*, *Bagel* greatly improves the robustness of deep generative models in KPI anomaly detection, by using CVAE to incorporate time information and dropout layer to avoid overfitting. Our experiments using real data from Internet companies show that, compared to *Donut*, *Bagel* improves the anomaly detection best F1-score by 0.08 to 0.43. Deep generative in general and VAE in specific are not sequential models. By successfully incorporating time information using CVAE to KPI anomaly problem, we believe we make an important step forward in making deep generative models work on more sequential scenarios.

ACKNOWLEDGMENT

We thank Haowen Xu for his helpful suggestions. The work was supported by National Natural Science Foundation of China (NSFC) under grant No. 61472214 and No. 61472210.

REFERENCES

- [1] S. Zhang, Y. Liu, D. Pei, Y. Chen, X. Qu, S. Tao, Z. Zang, X. Jing, and M. Feng, "Funnel: Assessing software changes in web-based services," *IEEE Transactions on Services Computing*, vol. 11, no. 1, pp. 34–48, 2018.
- [2] Y. Sun, Y. Zhao, Y. Su, D. Liu, X. Nie, Y. Meng, S. Cheng, D. Pei, S. Zhang, X. Qu *et al.*, "Hotspot: Anomaly localization for additive kpis with multi-dimensional attributes," *IEEE Access*, vol. 6, pp. 10909–10923, 2018.
- [3] F. Knorn and D. J. Leith, "Adaptive kalman filtering for anomaly detection in software appliances," in *INFOCOM Workshops 2008, IEEE*. IEEE, 2008, pp. 1–6.
- [4] B. Pincombe, "Anomaly detection in time series of graphs using arma processes," *Asor Bulletin*, vol. 24, no. 4, p. 2, 2005.
- [5] Y. Chen, R. Mahajan, B. Sridharan, and Z.-L. Zhang, "A provider-side view of web search response time," in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4. ACM, 2013, pp. 243–254.
- [6] H. Yan, A. Flavel, Z. Ge, A. Gerber, D. Massey, C. Papadopoulos, H. Shah, and J. Yates, "Argus: End-to-end service anomaly detection and localization from an isp's point of view," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 2756–2760.
- [7] W. Lu and A. A. Ghorbani, "Network anomaly detection based on wavelet analysis," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, p. 4, 2009.
- [8] N. Laptev, S. Amizadeh, and I. Flint, "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, 2015, pp. 1939–1947.
- [9] D. Liu, Y. Zhao, H. Xu, Y. Sun, D. Pei, J. Luo, X. Jing, and M. Feng, "Opprentice: Towards practical and automatic anomaly detection through machine learning," in *Proceedings of the 2015 Internet Measurement Conference*. ACM, 2015, pp. 211–224.
- [10] Z. Ding and M. Fei, "An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window," *IFAC Proceedings Volumes*, vol. 46, no. 20, pp. 12–17, 2013.
- [11] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng *et al.*, "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," in *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2018, pp. 187–196.
- [12] J. Bu, Y. Liu, S. Zhang, W. Meng, Q. Liu, X. Zhu, and D. Pei, "Rapid deployment of anomaly detection models for large number of emerging kpi streams," in *2018 IEEE 36th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2018, pp. 1–8.
- [13] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
- [14] M. Amer, M. Goldstein, and S. Abdennadher, "Enhancing one-class support vector machines for unsupervised anomaly detection," in *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*. ACM, 2013, pp. 8–15.
- [15] M. Sölk, J. Bayer, M. Ludersdorfer, and P. van der Smagt, "Variational inference for on-line anomaly detection in high-dimensional time series," *stat*, vol. 1050, p. 23, 2016.
- [16] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE*, vol. 2, pp. 1–18, 2015.
- [17] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *Proceedings of the 31st International Conference on International Conference on Machine Learning-Volume 32*. JMLR. org, 2014, pp. II–1278.
- [18] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *stat*, vol. 1050, p. 1, 2014.
- [19] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org>.
- [20] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Advances in Neural Information Processing Systems*, 2014, pp. 3581–3589.
- [21] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in Neural Information Processing Systems*, 2015, pp. 3483–3491.
- [22] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.
- [23] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie, "High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning," *Pattern Recognition*, vol. 58, pp. 121–134, 2016.
- [24] R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda, "Mawilab: combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking," in *Proceedings of the 6th International Conference*. ACM, 2010, p. 8.
- [25] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen, "Sketch-based change detection: methods, evaluation, and applications," in *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*. ACM, 2003, pp. 234–247.
- [26] R. Laxhammar, G. Falkman, and E. Sviestins, "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, 2009, pp. 756–763.
- [27] S.-B. Lee, D. Pei, M. Hajiaghayi, I. Pefkianakis, S. Lu, H. Yan, Z. Ge, J. Yates, and M. Kosseifi, "Threshold compression for 3g scalable monitoring," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 1350–1358.
- [28] A. Mahimkar, Z. Ge, J. Wang, J. Yates, Y. Zhang, J. Emmons, B. Huntley, and M. Stockert, "Rapid detection of maintenance induced changes in service performance," in *Proceedings of the Seventh Conference on emerging Networking EXperiments and Technologies*. ACM, 2011, p. 13.
- [29] M. Nicolau, J. McDermott *et al.*, "One-class classification for anomaly detection with kernel density estimation and genetic programming," in *European Conference on Genetic Programming*. Springer, 2016, pp. 3–18.
- [30] S. Shanbhag and T. Wolf, "Accurate anomaly detection through parallelism," *IEEE network*, vol. 23, no. 1, pp. 22–28, 2009.
- [31] A. H. Yaacob, I. K. Tan, S. F. Chien, and H. K. Tan, "Arima based network anomaly detection," in *Communication Software and Networks, 2010. ICCSN'10. Second International Conference on*. IEEE, 2010, pp. 205–209.
- [32] S. Zhang, Y. Liu, D. Pei, Y. Chen, X. Qu, S. Tao, and Z. Zang, "Rapid and robust impact assessment of software changes in large internet-based services," in *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*. ACM, 2015, p. 2.
- [33] M. Ma, S. Zhang, D. Pei, X. Huang, and H. Dai, "Robust and rapid adaption for concept drift in software system anomaly detection," in *The 29th IEEE International Symposium on Software Reliability Engineering (ISSRE 2018)*. IEEE, 2018, pp. 1–12.
- [34] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," 2018.