High-dimensional Nonlinear Profile Monitoring based on Deep Probabilistic Autoencoders

Nurettin Sergin¹, Hao Yan¹

¹School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ 85281 USA

Abstract

Wide accessibility of imaging and profile sensors in modern industrial systems created an abundance of high-dimensional sensing variables. This led to a a growing interest in the research of high-dimensional process monitoring. However, most of the approaches in the literature assume the in-control population to lie on a linear manifold with a given basis (i.e., spline, wavelet, kernel, etc) or an unknown basis (i.e., principal component analysis and its variants), which cannot be used to efficiently model profiles with a nonlinear manifold which is common in many real-life cases. We propose deep probabilistic autoencoders as a viable unsupervised learning approach to model such manifolds. To do so, we formulate nonlinear and probabilistic extensions of the monitoring statistics from classical approaches as the expected reconstruction error (ERE) and the KL-divergence (KLD) based monitoring statistics. Through extensive simulation study, we provide insights on why latent-space based statistics are unreliable and why residual-space based ones typically perform much better for deep learning based approaches. Finally, we demonstrate the superiority of deep probabilistic models via both simulation study and a real-life case study involving images of defects from a hot steel rolling process.

Note to Practitioners

This paper investigates whether deep probabilistic autoencoders can improve process monitoring of high-dimensional data such as images. Our motivating example consist of images collected from a hot steel rolling process with various types of defects. The goal is to detect when the system starts producing unseen defects. Existing methods fail to fully address the nonlinearity of the latent structure of in-control samples. We demonstrated both on simulated and real-life dataset that deep learning methods can recover the structures of HD data via few latent variables. We also demonstrate that the statistics based on residual space should be used. Finally, we provide a guideline on how to optimize hyperparameters for such a scenario where we don't have access to out-of-control data, which is crucial when applying this method in real systems.

1 Introduction

In recent years; variability, affordability, and ubiquity of imaging and profile sensors in modern industrial systems create an abundance of high-dimensional (HD) profiles. HD profile monitoring, refers to the procedure to quickly detect the change of HD profiles is an important problem in modern industrial systems. In literature, profiles can be defined as sensing variables that form a functional relationship of response variables with a set of factors of an experiment.

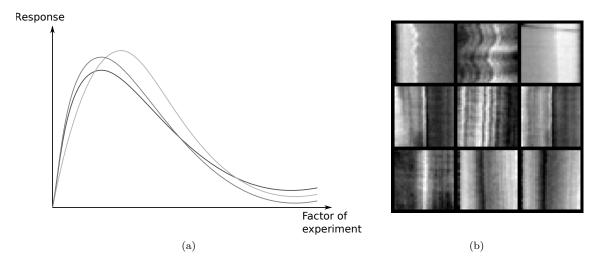


Figure 1: Generalized depiction of 1D profiles (a) and a real-life examples of 2D profiles from a hot steel rolling process (b).

For some examples, Figure 1a presents a generalized depiction of 1D profiles commonly found in literature. Figure 1b depicts images (i.e., 2D profiles) acquired from a hot steel rolling process, providing a typical example of HD profile. Each image in the example has more than two thousand pixels.

A common way to tackle the problem of profile monitoring is to assume a parametric regression profile. Parametric profile monitoring techniques model the relationship between the explanatory variables and response using a parametric function. Process monitoring techniques can then be used on the coefficients of this function. For example, linear profile monitoring assumes that the profile can be represented by a simple linear model and the slope and intercept can be monitored [1]. Zou et al. utilized a multivariate linear regression model for profiles with the LASSO penalty and used the regression coefficients for Phase-II monitoring [2]. However, linearity assumption often does not hold for the profile data. To address this challenge, nonlinear parametric models are proposed [3–6]. A major drawback of parametric profile monitoring techniques is that they require the parametric functions to be known, which can be unrealistic for complex profiles.

Another large body of profile monitoring research focuses on the type of profile data where the basis of the representation is assumed to be known but the coefficients are unknown. For instance, to monitor smooth profiles, various non-parametric methods based on local kernel regression [7–9] and splines [10] have been developed. To monitor the non-smooth wave-form signals, wavelet-based mixed effect model is proposed [11]. However, for all the aforementioned methods, it is assumed that the nonlinear variation pattern of the profile is well captured by a set of known basis or kernel. Usually, there is no guidance on selecting the right basis of the representation for the original data and it requires much trial and error.

Dimensionality reduction techniques have been studied to overcome the challenge of unknown parametric form or unknown basis representation. These techniques aim to learn the bases and/or lower-dimensional representations in a data-driven fashion. Principal component analysis (PCA) is the most popular method in this context for profile data monitoring due to its simplicity, scalability, and good data compression capability. In an example work, Liu [12] proposed PCA to reduce the

dimensionality of the streaming data and constructed the so-called T^2 and Q charts to monitor the extracted features and residuals, respectively. To generalize PCA methods to monitor the complex correlation for multi-channel profiles, Paynabar et al [13] proposed a multivariate functional PCA method and applied a change point detection on the functional coefficient. Along this line, tensor-based PCA methods are also proposed for multi-channel profiles such as uncorrelated multilinear PCA [14] and multi-linear PCA [15]. Finally, various tensor-based PCA methods [16] have been compared and different test statistics are developed for tensor-based process monitoring. The main limitation of these PCA related methods is the limited expressive power of linear features due to restricted profiles represented by a linear combination of the low-dimensional loadings set. Furthermore, each principal component represents a global variation pattern of the original profiles, which fails to capture the local spatial correlation within each single profile. In some cases, the model might force a solution with a much larger latent space than the actual number of latent space, yielding a sub-optimal and overfitting-prone representation, which will hinder the performance of process monitoring. A systematic discussion of this issue is articulated in [17]. In that work, the authors identify the problems associated with assuming closeness relationship in the subspace that is characterized by Euclidean metrics. They successfully observe that the sample-to-sample variation in complex high-dimensional corpora may lie on a nonlinear manifold. The variation semantics of nonlinear latent space can be quite different from the linear latent space. We attempt to provide a conceptual example in Figure 2, where Figure 2b represents a case in which the assumptions doesn't hold and two points on the latent space (p and q) will probably end up being much closer to each other than they actually are under a linear dimensionality reduction framework. However, Shi et al [17] only focus on applying manifold learning to model the nonlinear variation of the nonlinear profile, where the process monitoring procedure is not defined.

Recently, deep learning based solutions have been applied to certain data-driven tasks and achieved great success [18, 19]. Stacking a set of user-defined layers in the hypothesis space helped such models to learn highly nonlinear mappings [20]. Architectural advances in deep learning helped injecting some domain knowledge into the models. A good example is convolutional neural networks [19] which have achieved great success in tasks such as image recognition [18]. Convolutional operations help the model account for signal/image semantics such as the translational invariance. Convolutional architecture can be efficiently computed on GPU, which satisfies the online requirement for real-time monitoring. Given these advantages, deep dimensionality reduction models can provide a great alternative to classical dimensionality reduction techniques. In fact, deep autoencoders have been proposed for profile monitoring in [21] for Phase-I analysis, initiating a possible trend. Yan et al. [22] compared the performance of contractive autoencoders and denoising auto encoders for Phase-II monitoring. Zhang et al. [23] proposed a denoising autoencoder for process monitoring. Later, probabilistic autoencoders have been proposed for Phase-II analysis; for example, Zhang et al. [24] proposed a variational autoencoder with monitoring statistics in the latent space. However, we will demonstrate in this paper that considering only the monitoring statistics in the latent space is not sufficient. Despite these initial works, the monitoring statistics for different types of probabilistic autoencoders are not systematically defined, and the link with the traditional PCA-based monitoring techniques are not well studied.

In this paper, we propose two monitoring statistics—the expected reconstruction error (Q_{ERE}) and the KL-divergence (T_{KLD}^2) —for general probabilistic autoencoders. We illustrate the usage of these statistics for two popular deep probabilistic autoencoders, Variational Autoencoders (VAE) [25], and Adversarial Autoencoders (AAE) [26], for the task of monitoring of high-dimensional nonlinear profiles. Specifically, our work makes the following contributions:

• We study how the proposed $Q_{\rm ERE}$ and $T_{\rm KLD}^2$ statistics are a natural nonlinear probabilistic extension for autoencoders of the traditional T^2 and Q charts of PCA-based process monitor-

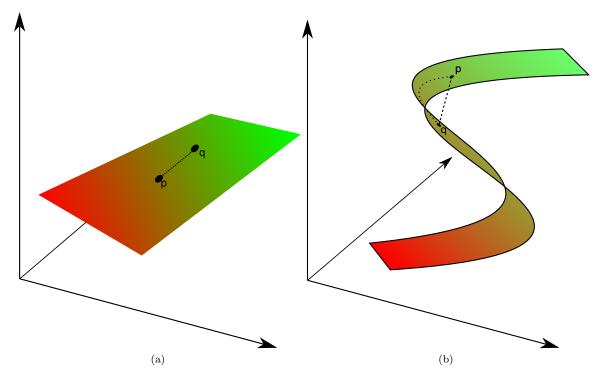


Figure 2: Illustrations of linear (a) and nonlinear (b) 2D subspaces in 3D. For (a), Euclidean and geodesic distance are overlapping and depicted with a dotted line between point q and p. In (b), both geodesic and Euclidean distances are depicted. The coloring is to aid representation of geodesic closeness of the points on the subspace.

ing.

- We provide important insight into why latent dimension based charts cannot perform well on the task of process shift detection given probabilistic autoencoder models by conducting extensive simulation study with controlled latent structure.
- We give practical advise with empirical support on how to tune hyperparameters for deep learning based models for the purpose of fault detection.
- We verify our results on a real-life case study based on a hot steel rolling image inspection systems.

The rest of the article is organized as follows: Section 2 introduces the reader to three deep autoencoding models. Section 3 outlines the procedure for setting up the process monitoring and what monitoring statistic we use. Sections 4 and 5 presents our findings on simulated and real-life data, respectively. Finally, we conclude our work in Section 6.

2 Background

We begin this section by making a formal definition of the general autoencoder.

	Deterministic	Probabilistic
Linear	PCA	PPCA
Nonlinear	AE	VAE, AAE

Definition 2.1. An autoencoder consists of two separate but closely related functions, the encoder g_{ϕ} and the decoder f_{θ} , parametrized by parameter vectors ϕ and θ respectively. Given a set of points $\mathcal{D} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)} \dots \mathbf{x}^{(n)}\}$ in \mathbb{R}^d , the encoder $g_{\phi} : \mathbb{R}^d \to \mathbb{R}^r$ is a mapping from the high-dimensional space d to a chosen low dimensionality r < d, while $f_{\theta} : \mathbb{R}^r \to \mathbb{R}^d$ is a mapping from the low-dimensional space to the high-dimensional space.

An autoencoder can be classified in several ways. First, autoencoders can be classified as linear and nonlinear, based on whether the encoder and decoder apply a linear or nonlinear transformation. Second, autoencoders can be classified as deterministic or probabilistic depending on whether g_{ϕ} and f_{θ} represent deterministic functions or stochastic distributions. For the probabilistic autoencoders, g_{ϕ} and f_{θ} are defined as conditional probability distribution, which is often denoted as $q_{\phi}(\mathbf{z} \mid \mathbf{x})$ and $p_{\theta}(\mathbf{x} \mid \mathbf{z})$ for clarity.

Autoencoders are very powerful for tasks such as denoising and manifold learning. They are also shown to increase the performance of supervised learning tasks when used as an unsupervised pretraining step [27]. The main idea behind the autoencoder models lies in the fact that most of the information in high-dimensional space is redundant and intrasample variability lies on a low-dimensional manifold. If these assumptions hold, the encoder can map the data to the low-dimensional manifold while the decoder can reconstruct the data from that low-dimensional manifold without significant information loss.

Section 2 shows some examples of popular autoencoders. Here, PCA is typically considered as a linear autoencoder. In contrast, models such as neural network based autoencoders are considered as nonlinear autoencoders. Similarly, probabilistic PCA (PPCA) typically falls under the category of the probabilistic linear autoencoders. This paper will focus on the probabilistic and nonlinear autoencoders. In the remaining of this section, we will review two major types of autoencoders, including Deterministic Autoencoders and Probabilistic Autoencoders. More specifically, we will walk through the following autoencoders: Deterministic Autoencoders (AE), Variational Autoencoders (VAE) and Adversarial Autoencoders (AAE). Specifically, we will discuss the commonality of these methods.

2.1 Deterministic Autoencoders

The first extensive explanation of a neural network based autoencoder is done in [28]. The basic idea is that the encoder g_{ϕ} is a neural network with parameters ϕ and the decoder is another neural network f_{θ} with parameters θ . The reconstruction of each sample point $\mathbf{x}^{(i)} \in \mathcal{D}$ is done by successively pushing the point through the two networks and expect a result that is similar to the original point $\mathbf{x}^{(i)} \approx g_{\phi}\left(f_{\theta}(\mathbf{x}^{(i)})\right)$. The network is trained to optimize the parameters, θ and ϕ , via back-propagation and stochastic gradient descent. The vanilla autoencoder has a simple loss function which can be formally defined as:

$$L_{AE}(\theta, \phi) = \frac{1}{n} \sum_{i=1}^{n} \left(\mathbf{x}^{(i)} - g_{\phi} \left(f_{\theta}(\mathbf{x}^{(i)}) \right) \right)$$
 (1)

Vanilla autoencoders may suffer from overfitting. Stacked denoising autoencoders and contractive autoencoders are presented as a form of regularization to combat the overfitting [27,29]. We will use

the denoising variant of autoencoders as our benchmark method by simply adding a dropout layer at the very beginning of the network.

2.2 Probabilistic Autoencoders

Probabilistic autoencoders replace deterministic functions g_{ϕ} and f_{θ} with conditional probability distribution $q_{\phi}(\mathbf{z} \mid \mathbf{x})$ and $p_{\theta}(\mathbf{x} \mid \mathbf{z})$ in both VAE [25] and AAE [26].

Both VAE and AAE assume a standard Gaussian prior on the latent variables $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The prior penalizes the use of complicated latent code distribution to represent the data distribution, which regularizes the complexity of the models. Both models assume that the decoder function follows a normal distribution with isotropic (spherical) covariance as $p_{\theta}(\mathbf{x} | \mathbf{z}) = \mathcal{N}(g_{\phi}(\mathbf{z}), \sigma^2 \mathbf{I})$. Here, the mean of reconstruction is modeled as another network, the decoder $g_{\phi}(\mathbf{z})$. Furthermore, both models approximate the actual posterior $q^*(\mathbf{z}|\mathbf{x})$ with a probabilistic encoder function as a Gaussian distribution $q_{\phi}(\mathbf{z} | \mathbf{x}) = \mathcal{N}(\mu(\mathbf{x}), \operatorname{diag}(\sigma(\mathbf{x}))^2)$ for tractibility purposes. Here, $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ together constitute the encoder network that infers the mean and the variance of the posterior distribution, respectively.

VAE and AAE differ on how they impose the prior regularization. VAE uses the variational inference to optimize the parameters whereas AAE uses adversarial training. VAE uses a tractable lowerbound on the intractable log-likelihood of the data $\log(p(\mathbf{X}))$, defined as the evidence lower bound (ELBO) and formulated as follows:

$$L_{ELBO} = \log (p(\mathbf{x})) - \text{KL} (q_{\phi}(\mathbf{z} \mid \mathbf{x}) \parallel q^{*}(\mathbf{z} \mid \mathbf{x}))$$

$$= \mathbb{E}_{\mathbf{z} \sim q_{\phi}(\mathbf{z} \mid \mathbf{x})} \log p_{\theta}(\mathbf{x} \mid \mathbf{z}) + \text{KL} (q_{\phi}(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})),$$
(2)

where KL ($\cdot \parallel \cdot$) denotes the Kullback-Leibler divergence between two distributions. From Equation (2), we can see that L_{ELBO} is a lower-bound of the log-likelihood function $\log{(p(\mathbf{x}))}$ since Kullback-Leibler divergence is always nonnegative. The first component of the right-hand side $\mathbb{E}_{\mathbf{z} \sim q_{\phi}(\mathbf{z} \mid \mathbf{x})} \log p_{\theta}(\mathbf{x} \mid \mathbf{z})$ is the expected log-likelihood, which can be well approximated by Monte Carlo sampling.

$$\mathbb{E}_{\mathbf{z} \sim q_{\phi}(\mathbf{z} \mid \mathbf{x})} \log p_{\theta}(\mathbf{x} \mid \mathbf{z}) \approx \frac{1}{m\sigma^{2}} \sum_{i=1}^{m} \|\mathbf{x} - f_{\theta}(\mathbf{z}_{i})\|^{2}$$
where $\mathbf{z}_{i} \sim q_{\phi}(\mathbf{z} \mid \mathbf{x})$ (3)

The second term KL $(q_{\phi}(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z}))$ is the KLD term between a diagonal multivariate Gaussian distribution and standard Gaussian distribution, which can be calculated via Equation (4). The reader can refer to the original paper [25] for the detailed derivation of the VAE framework.

$$KL\left(\mathcal{N}\left(\mu(\mathbf{x}), \operatorname{diag}(\sigma(\mathbf{x}))\right) \parallel \mathcal{N}(\mathbf{0}, \mathbf{I})\right)$$

$$= \frac{1}{2} \sum_{i=1}^{r} \left(\mu_i(\mathbf{x})^2 + \sigma_i(\mathbf{x})^2 - \log(\sigma_i(\mathbf{x})^2) - 1\right)$$
(4)

On the other hand, AAE takes a different approach to regularize the prior distribution of \mathbf{z} via another discriminator network D, which tries to guess whether a given latent code is coming from the actual prior distribution $p(\mathbf{z})$ or the posterior distribution, estimated by the encoder network $q_{\phi}(\mathbf{z} \mid \mathbf{x})$. Formally, the discriminator tries to maximize the objective function given in Equation (5).

$$L_D = \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} \log(1 - D(\mathbf{z})) + \mathbb{E}_{\mathbf{z} \sim q_{\phi}(\mathbf{z} \mid \mathbf{x})} \log(D(\mathbf{z}))$$
 (5)

Concurrently, the encoder is trained to fool the discriminator while also minimizing the reconstruction error working together with the decoder. Through a minimax game between the encoder and the discriminator networks, the encoder gradually learns to output latent codes that resembles $p(\mathbf{z})$.

$$\min_{\theta, \phi} \max_{D} \mathbb{E}_{z \sim q_{\phi}(\mathbf{z} \mid \mathbf{x})} \log p_{\theta}(\mathbf{x} \mid \mathbf{z}) + L_{D}$$
(6)

This results in a Gaussian manifold learned from the data. Both AAE and VAE thrives to resemble the original distribution of $\mathbf{x} \sim P(\mathbf{X})$ through the latent code \mathbf{z} .

Gradient descent algorithms are often used to optimize these objective functions such as Equation (2) for VAE and Equation (6) for AAE. For example, the weights of the encoder, decoder, and the discriminator (in the case of AAE) are jointly optimized via the back-propagation and gradient descent. We aim to reveal these common structures among the models we propose. The detailed training procedures are discussed in details in the original VAE [30] and AAE papers [31]. For more discussion for the training procedure of the adversarial learning methods (e.g., AAE) please refer to [32].

3 Proposed Monitoring Statistics and Monitoring Procedure

In this section, we first review the traditional T^2 ad Q control chart statistics for PCA methods in Section 3.1. Motivated by this, we will then propose two monitoring statistics, namely the ERE and KLD control chart for profile monitoring based on general probabilistic autoencoders. We will also discuss why $Q_{\rm ERE}$ and $T_{\rm KLD}^2$ are natural probabilistic extensions of T^2 and Q for PCA.

3.1 Review of T^2 and Q statistics

Process monitoring via PCA typically define two monitoring statistics, namely the T^2 and Q statistics [33]. The Q statistic for PCA is defined as the reconstruction error between the real sample \mathbf{x} and the reconstructed sample $\tilde{\mathbf{x}}$. The geometric representation is to measure how far the sample is away from the learned subspace of in-control (IC) samples. T^2 represents the how far the sample is away from the center of latent codes of the IC samples.

The T^2 and Q statistics for PCA are defined as follows:

$$Q(\mathbf{x}) = \|\mathbf{x} - \tilde{\mathbf{x}}\|^2 = \|\mathbf{x} - \mathbf{W}\mathbf{W}^{\top}\mathbf{x}\|$$

$$T_{PCA}^2(\mathbf{x}) = \mathbf{z}^{\top}\mathbf{\Sigma}^{-1}\mathbf{z} = \mathbf{x}^{\top}\mathbf{W}\mathbf{\Sigma}^{-1}\mathbf{W}^{\top}\mathbf{x},$$
(7)

where matrix **W** is the loading matrix, and Σ^{-1} is the inverse of the covariance matrix of the latent code **z**.

3.2 Proposed Q_{ERE} and T_{KLD}^2 charts for probabilistic autoencoders

In this subsection, we propose the ERE and KLD charts to extend the T^2 -chart and Q-chart for probabilistic encoders and decoders. First, to extend the Q-statistic to the probabilistic setting, we propose the expected reconstruction error, denoted by $Q_{\rm ERE}$ to show the link between the traditional Q-chart, which can be computed in Equation (8).

$$Q_{ERE} = E_{\mathbf{z} \sim q_{\phi}(\mathbf{z} \mid \mathbf{x})} \log p_{\theta}(\mathbf{x} \mid \mathbf{z}). \tag{8}$$

Second, to extend the T^2 statistics into the probabilistic setting, we propose to define the KLD statistic, denoted by T_{KLD}^2 , as the distance of the latent posterior $q_{\phi}(\mathbf{z} \mid \mathbf{x})$ and the prior distribution $p(\mathbf{z})$ (e.g., the IC distribution in the empirical Bayesian framework) as Equation (9)

$$T_{\text{KLD}}^2 = \text{KL}\left(q_{\phi}(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})\right) \tag{9}$$

The proposed Q and T^2 statistics can be generalized for other common distributions selected for prior $p(\mathbf{z})$, $q_{\phi}(\mathbf{z} | \mathbf{x})$, and $p_{\theta}(\mathbf{x} | \mathbf{z})$. However, in this paper, we will focus on deriving the formulation based on the Gaussian distribution assumption outlined in Proposition 1. Especially, we will focus on the standard normal prior for the latent variable \mathbf{z} , which is a common assumption for Probabilistic PCA (PPCA) [34], VAE, and AAE. The reason for such an assumption is that the latent code \mathbf{z} is typically assumed to be uncorrelated and the scaling will be handled in the transformation function.

Proposition 1. If the prior, encoding and decoding functions are normally distributed as:

$$P(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$$

$$p_{\theta}(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}(f_{\theta}(\mathbf{z}), \sigma^{2}\mathbf{I})$$

$$q_{\phi}(\mathbf{z} \mid \mathbf{x}) = \mathcal{N}(\mu(\mathbf{x}), \operatorname{diag}(\sigma(\mathbf{x})))$$

Then T_{KLD}^2 becomes:

$$T_{\text{KLD}}^{2} = \text{KL}\left(\mathcal{N}\left(\mu(\mathbf{x}), \text{diag}(\sigma(\mathbf{x}))\right) \parallel \mathcal{N}(\mathbf{0}, \mathbf{I})\right)$$

$$= \frac{1}{2} \sum_{i=1}^{r} \left(\mu_{i}(\mathbf{x})^{2} + \sigma_{i}(\mathbf{x})^{2} - \log(\sigma_{i}(\mathbf{x})^{2}) - 1\right)$$
(10)

Likewise, Q_{ERE} can be derived as:

$$Q_{\text{ERE}} \propto E_{\mathbf{z} \sim q_{\phi}(\mathbf{z} \mid \mathbf{x})} \|\mathbf{x} - f_{\theta}(\mathbf{z})\|^{2}$$
(11)

The proof for T_{KLD}^2 follows from [35, p. 13]. For Q_{ERE} , it follows simply from the definition of multivariate Gaussian density, such that the formulation is equivalent to Equation (8) up to a constant. Finally, Equation (11) can often be well approximated by using the reconstruction error of the posterior mean as $Q_{\text{ERE}} \approx \|\mathbf{x} - f_{\theta}(\mu(\mathbf{x}))\|^2$, when the variance of the posterior distribution is small.

To relate the proposed $Q_{\rm ERE}$ and $T_{\rm KLD}^2$ with the traditional T^2 and Q-chart, we will derive the proposed $Q_{\rm ERE}$ and $T_{\rm KLD}^2$ in the case of linear probabilistic autoencoders as the PPCA in Proposition 1

Proposition 2. Besides the Gaussian assumption in Proposition 1, PPCA further assumes that the decoder is a linear transformation $p_{\theta}(\mathbf{x} | \mathbf{z}) = \mathcal{N}(\mathbf{Wz}, \sigma^2 \mathbf{I})$ over the prior $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$. In this case, the encoder becomes $q_{\phi}(\mathbf{z} | \mathbf{x}) = q^*(\mathbf{z} | \mathbf{x}) = \mathcal{N}(\mathbf{M}^{-1}\mathbf{W}^{\top}\mathbf{x}, \mathbf{M}^{-1})$, where $\mathbf{M} = \mathbf{W}^{\top}\mathbf{W} + \sigma^2 \mathbf{I}$ as explained in [34, p. 614]. In this case, we define T_{KLD}^2 and Q_{ERE} for PPCA as:

$$T_{\text{KLD}}^{2}(\mathbf{x}) = \|\mu(\mathbf{x})\|^{2} \tag{12}$$

$$Q_{\text{ERE}}(\mathbf{x}) = E_{\mathbf{z} \sim q_{\phi}(\mathbf{z} \mid \mathbf{x})} \| \mathbf{x} - \mathbf{W} \mathbf{z} \|^{2}$$
(13)

The test statistic T_{KLD}^2 is identical to the T^2 statistic for PCA as defined in Equation (7). In fact, if we let $\sigma \to 0$, PPCA becomes the standard PCA method, and the $q_{\phi}(\mathbf{z} \mid \mathbf{x})$ becomes the Dirac Delta function, as $q_{\phi}(\mathbf{z} \mid \mathbf{x}) = \delta(\mathbf{z} - \mathbf{W}^{\top} \mathbf{x})$. In this case, $Q_{\text{ERE}}(\mathbf{x}) \propto \|\mathbf{x} - \mathbf{W} \mathbf{W}^{\top} \mathbf{x}\|^2$ becomes the Q statistic for PCA as defined in Equation (7).

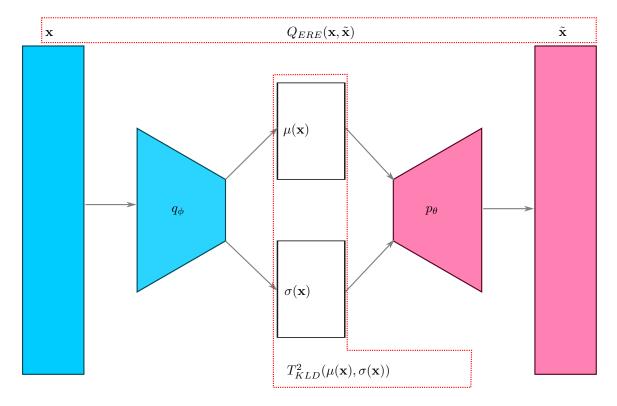


Figure 3: Graphical depiction of the proposed monitoring statistics with probabilistic autoencoders

Decomposability of the Proposed Monitoring Statistics

 T_{KLD}^2 in Equation (10) is decomposable since we are using a diagonal covariance structure $q_{\phi}(\mathbf{z} \mid \mathbf{x}) = \prod_{i \in \{1...r\}} q(z_i \mid \mathbf{x})$ and standard Gaussian as prior $p(\mathbf{z}) = \prod_{i \in \{1...r\}} p(z_i)$. Precisely, T_{KLD}^2 can be

decomposed as $T_{\text{KLD}}^2 = \sum_{i \in \{1...r\}} \text{KL}\left(q(z_i|\mathbf{x}) \parallel p(z_i)\right)$. Similarly, the Q_{ERE} can also be decomposed to different dimensions given that we assume an isotropic error structure for $p_{\theta}(\mathbf{x} | \mathbf{z})$. As long as $p_{\theta}(\mathbf{x} | \mathbf{z}) = \prod_{i \in \{1,...d\}} p(x_i | \mathbf{z})$, we can decompose Q_{ERE} as $Q_{\text{ERE}} = E_{z \sim q_{\phi}(\mathbf{z} \mid \mathbf{x})} \log p_{\theta}(\mathbf{x} \mid \mathbf{z}) = \sum_{i \in \{1...d\}} E_{z \sim q_{\phi}(\mathbf{z} \mid \mathbf{x})} \log p(x_i \mid \mathbf{z}).$ The decomposability of T_{KLD}^2 and Q_{ERE} is important as it might help in tracing back which

dimension of the latent code z or the original data x is responsible when an OC sample is detected.

3.4 **Profile Monitoring Procedure**

A typical profile monitoring procedure follows two processes, Phase-I analysis and Phase-II analysis. Phase-I analysis results in a trained model (i.e. an encoder and a decoder) and an Upper Control Limit (UCL) to help setup the control chart for each of the monitoring statistics. In Phase-II analysis, the system is exposed to new profile samples generated via the industrial setting and the model decides whether these samples are IC or out-of-control (OC). The general procedure of the profile monitoring for deep autoencoders is defined as follows:

• Obtain IC dataset \mathcal{D} and partition it into train, validation and test sets \mathcal{D}^{trn} , \mathcal{D}^{val} , \mathcal{D}^{tst}

- Train an encoder f (or $q_{\phi}(\mathbf{z} | \mathbf{x})$) and a decoder g (or $p_{\theta}(\mathbf{x} | \mathbf{z})$) using samples from \mathcal{D}^{trn}
- Calculate test statistic for all $\mathbf{x} \in \mathcal{D}^{val}$ and take it's 95th percentile as the UCL.
- Record the estimated false alarm rate (FAR) as the average number of samples $x \in \mathcal{D}^{tst}$ that are misclassified as OC because their test statistic yields a number higher than UCL. This will be useful to test the robustness of the UCL.
- Start collecting sample profiles from the process and make decisions on whether they are OC or not based on UCL.

4 Simulation Study, Architecture, and Hyperparameter Tuning

4.1 Gasket Bead Simulation Setup

We first evaluate the performance of the deep autoencoding models in a simulation setting inspired by the work of Shi et al. [17]. The simulation procedure produces 2D point clouds, similar to the scanning of a gasket bead. There are two main sources of variation that affect the outcome of any sample: c_0 is the horizontal component of the center location of a bead, and a controls the width of the bead on the horizontal axis. We assume the vertical center c_1 and the vertical width b to be fixed at 0.5 and 0.2, respectively. An IC sample consists of a grid of tuples for which the following function is applied:

$$g(p_0, p_1; c_0, a) = 1 - \frac{(p_0 - c_0)^2}{a} - \frac{(p_1 - c_1)^2}{b}$$

$$f(p_0, p_1; c_0, a) = \begin{cases} \sqrt{g(p_0, p_1; c_0, a)} + \epsilon & \text{if } g(p_0, p_1; c_0, a) > 0\\ \epsilon & \text{if } g(p_0, p_1; c_0, a) < 0 \end{cases}$$

$$(14)$$

On both vertical and horizontal dimensions, the grid p_{0i} and p_{1j} are defined as an equally spaced array stretching from 0 to 1. For this study, we choose to create grids of size 64 by 64. Noise $\epsilon \sim \mathcal{N}(0, 0.01)$ is added per each pixel. The samples are best visualized as grayscale images as illustrated in Figure 4.

We define IC patterns by generating two independent normal distributions on the latent location c_0 and width a as follows:

$$c_0 \sim \mathcal{N}(0.5, 1 \times 10^{-2})$$

 $a \sim \mathcal{N}(0.2, 6.25 \times 10^{-4})$ (15)

Finally, we will consider the following four types of OC variation patterns for the system.

- For location shift, the mean of the process that generates c_0 is altered by an amount δ as $c_0 \leftarrow c_0 + \delta * 1 \times 10^{-1}$.
- For width shift, the mean of the process that generates a is perturbed by an amount δ as $a \leftarrow a + \delta * 2.5 \times 10^{-2}$.
- For mean shift, all pixel values shifts by a certain amount δ as $f(p_{0i}, p_{1j}; c_0, a) \leftarrow f(p_{0i}, p_{1j}; c_0, a) + \delta$.

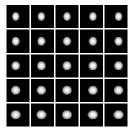


Figure 4: Illustrations of simulated gasket beads images. From left to right the horizontal component of center location c_0 shifts while from top to bottom horizontal width a increases

• For magnitude shift, all pixel values are multiplied by a certain factor δ , resembling an increase or decrease in the height of the bead as $f(p_{0i}, p_{1j}; c_0, a) \leftarrow f(p_{0i}, p_{1j}; c_0, a) * \delta$.

Here, δ is the intensity of the change, which varies from 0 to 3 for all changes. Location shift and width shift occur the latent space whereas mean shift and magnitude shift occur in the observed space.

To illustrate the power of deep probabilistic autoencoders even with a small sample size, the training, validation, and testing IC samples are generated of size 256 each, as well as OC samples of the same size with different intensity δ . We will repeat the experiments for 10 times with different seeds that generate the data with different hyperparameters.

4.2 Comparison Between T_{KLD}^2 and Q_{ERE} Statistics

When comparing the performance of $T_{\rm KLD}^2$ and $Q_{\rm ERE}$, we observe that $T_{\rm KLD}^2$ falls behind $Q_{\rm ERE}$ significantly. The reader is encouraged to observe Table 1 for each comparison between the two for all possible encompassing cases of possible OC scenarios and model matches. The bolded numbers represent the better-performing statistic for that specific model and scenario match. It is clear that none of the cases favor the $T_{\rm KLD}^2$ statistic. We would like to discuss why this is the case in two different scenarios: change in the observation space and change in the latent space.

For changes in the observation space, such as the magnitude shift and mean shift, there is a significant difference between the median $Q_{\rm ERE}$ and $T_{\rm KLD}^2$ detection accuracies for all change magnitudes. This is because in most of the cases, anomalous behavior in the observed space cannot lead to a meaningful representation in the latent space. For example, the latent dimensions (e.g., center location and horizontal width) of a blank black image are not meaningful. Therefore, $Q_{\rm ERE}$ can easily capture these changes whereas $T_{\rm KLD}^2$ cannot.

For changes in the latent space, such as location shift and width shift, $Q_{\rm ERE}$ still outperforms $T_{\rm KLD}^2$. This may seem counter-intuitive, since $T_{\rm KLD}^2$ is specifically designed to capture the change in the latent space. There are two major reasons why $T_{\rm KLD}^2$ may not work well, even for the change in the latent space:

1) Challenge of disentangled representation learning: In the literature, the phenomenon of recovering the true independent latent structures is called disentangled representation learning. Figure 5 presents a visual summary of encodings of the validation IC samples of a trained AAE model with latent dimension fixed to 2. Figure 5a and Figure 5b demonstrates that AAE model works very well for image reconstruction and prior regularization (e.g., latent dimensions that roughly follows

Table 1: Median detection accuracies for comparison between statistic types

	Anomaly	Magni	tude	Mean Shift		Width		Location Shift			
	Intensity Level	Low	High	Low	High	Low	Mid	High	Low	Mid	High
Model	Stat Type										
AAE	Q	1.00	1.00	1.00	1.00	0.16	0.48	0.81	0.22	0.59	0.91
	T^2	0.14	0.33	0.05	0.05	0.04	0.02	0.00	0.04	0.01	0.00
VAE	Q	0.15	1.00	0.10	1.00	0.16	0.48	0.82	0.21	0.56	0.89
	T^2	0.09	0.14	0.04	0.02	0.04	0.12	0.19	0.11	0.16	0.11
PCA	Q	0.07	0.31	0.54	1.00	0.10	0.34	0.71	0.25	0.62	0.93
	T^2	0.07	0.14	0.05	0.05	0.06	0.18	0.35	0.24	0.60	0.90

a standard normal distribution). However, as shown in Figure 5b and Figure 5f, the true features (e.g. location shift and width shift) in the latent space found by the AAE method are highly entangled. For example, the color changes are observed in a circular pattern rather than aligned with the actual axis. Learning the disentangled representation is still a very challenging problem for deep autoencoders [36], which leads to a bad detection power. For example, in Figure 5c, OC samples are close to the IC samples and the origin, suggesting that the value of $T_{\rm KLD}^2$ statistics are smaller than others. 2) Unable to extrapolate the data. Deep auto-encoders would require one to train the encoders and decoders to act functionally in the regions of training data. However, for a completely unseen objects that are outside the training regions, the encoders will still generate the latent features within the standard Gaussian distribution due to the prior regularization. For example, if we randomly selected OC samples with mean shift and magnitude shift and then plotted the latent codes in Figure 5d and Figure 5h. These plots illustrate that the latent codes still follow the standard normal distribution. Therefore, the test statistics defined in the latent domain would not work well.

While $T_{\rm KLD}^2$ are designed to work on the latent space, ERE methods work on the observation spaces, which rely on whether decoders can reconstruct the original HD data. Similar to encoders, decoders are not capable to reconstruct the data and therefore cannot create what they have not seen before. However, this is actually beneficial for the $Q_{\rm ERE}$ statistic because it will increase the reconstruction error for OC samples while maintaining a similar level for IC samples.

4.3 Results Comparison and Sensitivity Analysis

Table 1 can also be used to compare the performance of different models. In the case of magnitude shift, mean shift, and width shift, AAE and VAE outperforms PCA by a large margin. For example, at a low magnitude of the magnitude shift, AAE can achieve a detection power of 100%, whereas PCA only detects 7% of the OC samples. However, for the location shift, PCA methods outperforms the AAE by a slight margin. For example, in the mid change magnitude, PCA can detect 62% of the samples, where VAE and AAE can detect 56% and 59%, respectivly. The reason is that PCA uses the fully connected networks, whereas the VAE and AAE use mostly the convolutional layers. Since convolutional layers are invariant to the location shift, these convolutional layers generalize too well even for OC samples. However, given the last FC() layers used in AAE and VAE, these

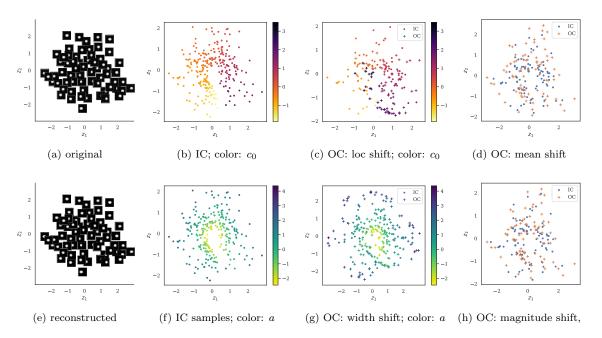


Figure 5: (a) and (e) shows original and reconstructed versions of a sample of IC gaskets taken from the test partition. Each image is positioned on the inferred mean locations $\mu(\mathbf{x})$. Coloring scheme in (b) and (c) shows how the actual center location c_0 changes over the inferred locations and coloring schemes in (f) and (g) show the change in actual width a. (c) and (g) depicts IC and OC samples together for comparison. (c) and (g) adds a mediocre case of OC samples on top of (b) and (f) respectively. (d) and (h) depicts the encodings for a mediocre case of mean shift and magnitude shift, respectively, as opposed to IC samples.

networks are still able to detect the location shift with similar performance.

Finally, we also perform the sensitivity analysis of how the proposed models are able to detect the OC samples with different change magnitude intensities summarized in Figure Figure 7. We can see that AAE gives the best overall performance out of the three models and AE generally yields a better performance than PCA. Also, advantage between the performances of AAE compared to PCA is much more apparent when the anomaly happens on the observed space such as mean shift and magnitude shift. This shows that AAE indeed learns the abstract representation and does not suffer from the overfitting, since it has a much larger reconstruction error applied to these OC samples.

4.4 Model Architectures & Implementation Details

It is challenging to specify a good model structure for the encoder and decoder with specific parametric form. Typically convolutional neural network is normally applied to learn the spatial correlation patterns,. For example, the encoders and decoders normally consist of convolutional layers, activation layers, and fully connected layers. The layers used in the papers are summarized as follows:

• C(O, K, S, P): Convolutional layer with arguments referring to number of output channels O, kernel size K, stride S and size of zero-padding P.

- CT(O, K, S, P): Convolutional transpose layer with arguments referring to the number of output channels O, kernel size K, stride S, and size of zero-padding P.
- \bullet FC(I, O): Fully connected layer with arguments referring to input dimension I and output dimension O respectively.
- R(): Rectified linear unit (i.e., ReLU) defined as $f(x) = \max(0, x)$
- LR(α): Leaky ReLU with single argument referring to negative slope α , which is defined as $f(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha x & \text{if } x < 0 \end{cases}$
- S(): Sigmoid function, defined as $f(x) = 1/(1 + \exp(-x))$.

The first three layers, C(), CT(), and FC() are considered the linear transformation layers. R(), $LR(\alpha)$, and S() are considered the nonlinear activation layers. Typically, C() with the stride can be used to decrease the spatial dimensions in the encoders. Pooling layers are typically not recommended in the autoencoders [37]. CT() layers can be used to increase spatial dimensions in the decoders. In VAE and AE, we use R() layers as activation layers. In AAE, we will use $LR(\alpha)$ as activation layers as suggested by [32]. Convolutional layers and activation layers are used alternatively until the last FC() layers.

For AAE, we must define the encoders, decoders and discriminators. An encoder will output 2r nodes, of which the first r nodes are used in the decoder to reconstruct the original image and the other r nodes are used for the input of the discriminator. For VAE, we need to define the probabilistic encoders and decoders. Encoder will output 2r latent features, which the first r variables are the parameters of the mean of the posterior distribution and the last r variables are the parameters of the standard deviation of the posterior distribution. For AE, we will simply map the output to r variables. Model architectures used for this study is summarized in Table 2. Dropout layers are used for the training phase, right before convolutional layers if the hyperparameter configuration dictates so. A grid of hyperparameters are considered for learning rate, dropout probability, batch sizes, latent dimensions, and the number of latent dimensions r. For AAE, the tradeoff weight between discriminator fooling loss and reconstruction loss is also considered as a hyperparameter. We discuss in the next subsection how to tune these hyperparameters.

4.5 Hyperparameter Tunning

One important question—especially from the practitioner's point-of-view—is how to decide which hyperparameter configuration to use. Unlike in a typical supervised learning task, we do not have access to OC data beforehand, leaving us without a clear objective to optimize. However, we leverage the fact that we are using only $Q_{\rm ERE}$ as the test statistic. We claim that the mean reconstruction error on a validation set is a good indicator of the detection power of deep probabilistic auto-encoders. Our reasoning is that probabilistic auto-encoders such as VAE and AAE have built-in regularization, therefore minimizing the reconstruction validation set will not lead to overfitting.

To test our hypothesis, we calculated the Pearson correlation coefficient between the validation reconstruction error and detection power for every specific model-anomaly-intensity combination, as well as each coefficient's significance level. If our hypothesis is correct, we expect a significant negative relationship between validation construction error and detection power. Our results, presented in Figure 6 overall support our claim. For all methods and various experiment settings, the correlation between the mean reconstruction error and detection accuracy is found to be negative for almost all replications. Also, roughly half of the p-values found this relationship to be significant.

Table 2: Architecture details of the models used in this study

Model Type	Sub Module	Architecture
AAE	Encoder	C(32, 4, 2, 1) - LR(0.2) - C(32, 4, 2, 1) - LR(0.2) - C(64, 4, 2, 1)
		- LR(0.2) - C(64, 4, 2, 1) - LR(0.2) - C(64, 4, 1, 0) - FC(256, $2r$)
	Decoder	FC(r, 256) - $LR(0.2)$ - $CT(64, 4, 0, 0)$ - $LR(0.2)$ - $CT(64, 4, 2, 1)$
		- LR(0.2) - C(32, 4, 2, 1) - CT(32, 4, 2, 1) - LR(0.2) - C(1, 4, 2, 1)
	Discriminator	FC(r, 512) - FC(512, 256) - FC(256, 1) - S()
VAE	Encoder	C(32, 4, 2, 1) - R() - C(32, 4, 2, 1) - R() - C(64, 4, 2, 1) - R()
		- C(64, 4, 2, 1) - R() - C(64, 4, 1, 0) - FC(256, $2r$)
	Decoder	FC(r, 256) - CT(64, 4, 0, 0) - R() - CT(64, 4, 2, 1) - R()
		- CT(32, 4, 2, 1) - R() - CT(32, 4, 2, 1) - R() - CT(1, 4, 2, 1)
AE	Encoder	C(32, 4, 2, 1) - R() - C(32, 4, 2, 1) - R() - C(64, 4, 2, 1) - R()
		- C(64, 4, 2, 1) - R() - C(64, 4, 1, 0) - FC(256, r)
	Decoder	FC(r, 256) - $CT(64, 4, 0, 0)$ - $R()$ - $CT(64, 4, 2, 1)$ - $R()$
		- CT(32, 4, 2, 1) - R() - CT(32, 4, 2, 1) - R() - CT(1, 4, 2, 1)

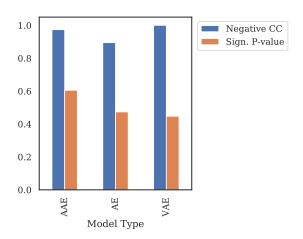


Figure 6: Ratio of negative valued correlation coefficients and the ratio of significant (≤ 0.05) 2-tailed p-values from Pearson's test

In comparison, the only hyperparameter for PCA is the number of principal components. Relying on validation error alone will force the model to choose as many principal components as possible. Choosing the right number of components is still an open problem and readers are referred to the related section of [38] for a review of alternatives. Here, we will use the number of components that keeps 99% of explained variation for PCA, which yields the best overall performance for gasket bead data for the subsequent performance comparisons such as the profile monitoring.

Given the advantage of using $Q_{\rm ERE}$ over $T_{\rm KLD}^2$, we will use a single test statistic— $Q_{\rm ERE}$ statistic—

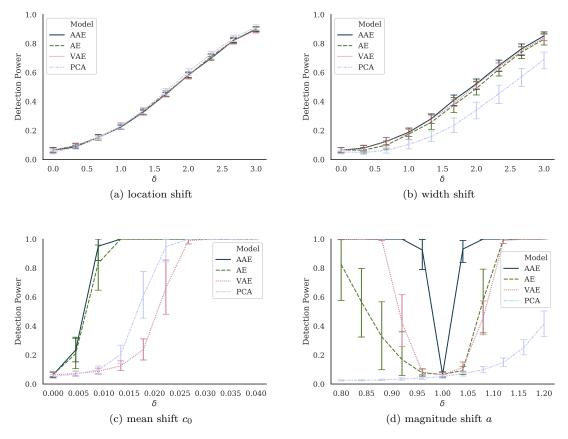


Figure 7: Detection power based comparison of AAE, AE, VAE and PCA for varying intensities of all OC behaviors. The error bars represents a 95% confidence interval

and a structured method to evaluate hyperparameter configurations for deep autoencoding models, we can make the final comparison of the performance of models. The results are based on ten replications of the same experiment where from one seed to another, the IC and OC data as well as the weight initializations of deep learning models differ. For each experiment, the hyperparameter configurations of deep learning models are based on the least validation reconstruction error.

5 Case Study and Results

In this section, we will use real images collected from a quality inspection in the rolling manufacturing process to illustrate the performance of the proposed process monitoring procedure. The dataset consists of metal rolling inspection images that are potentially defect. The domain engineers have labeled the images as normal or abnormal samples. Training data consists of 338 normal images. Ten different classes of 3552 defect images are also provided for performance testing. For every experiment, we randomly partition the IC corpus to train, validate and test with 60%-20%-20% relative sizes respectively. The rest of the procedure followed is outlined in Section 3. As mentioned

Table 3: Q_{ERE} Statistic Based Detection Power for OC Classes

Model Type OC Class #	AAE	AE	VAE	PCA
2	$0.81{\pm}0.10$	0.68 ± 0.29	0.70 ± 0.22	0.62 ± 0.05
3	$\boldsymbol{0.93} {\pm} \boldsymbol{0.03}$	$0.90 {\pm} 0.04$	$0.88 {\pm} 0.03$	$0.86{\pm}0.02$
4	$\boldsymbol{0.76 {\pm} 0.09}$	$0.76{\pm}0.15$	$0.76{\pm}0.11$	$0.67 {\pm} 0.07$
5	$\boldsymbol{1.00}{\pm0.00}$	$\boldsymbol{1.00 {\pm} 0.00}$	$\boldsymbol{1.00 {\pm} 0.00}$	$1.00{\pm}0.00$
6	$\boldsymbol{0.87} {\pm} \boldsymbol{0.05}$	$0.84{\pm}0.08$	$0.85{\pm}0.07$	$0.87{\pm}0.03$
7	$0.17{\pm}0.04$	$\boldsymbol{0.22 {\pm} 0.06}$	$\boldsymbol{0.22 {\pm} 0.07}$	$0.12 {\pm} 0.01$
8	$0.65{\pm}0.08$	$0.67{\pm}0.14$	$\boldsymbol{0.70 {\pm} 0.14}$	$0.45{\pm}0.06$
10	$\boldsymbol{0.83 {\pm}.0.09}$	$0.70 {\pm} 0.15$	$0.67{\pm}0.018$	$0.75 {\pm} 0.04$
12	$\boldsymbol{0.95} {\pm} \boldsymbol{0.04}$	$0.89 {\pm} 0.07$	$0.87{\pm}0.10$	$0.90 {\pm} 0.04$
13	$\boldsymbol{0.91} {\pm 0.04}$	$0.83 {\pm} 0.08$	$0.80 {\pm} 0.12$	$0.88 {\pm} 0.02$

before, we will use only $Q_{\rm ERE}$ statistic and find the best configuration for deep autoencoding models based on the smallest validation mean reconstruction error.

Overall, our results once again favor AAE overall against every other model, as can be seen from Table 3. Both AE and VAE perform worse than AAE. AE requires careful tune of the neural network architecture and typically results in large variation without proper regularization. For PCA, we have tried all possible PCs and picked the best one given the testing error, giving PCA some advantage over the rest architectures, where the architecture is not optimized using any testing data. Despite this advantage, PCA performed the worst out of the models. Finally, we acknowledge that the performance of deep autoencoding models can be significantly improved by data augmentation techniques and the search of neural network architecture guided by domain experts. We didn't consider the data augmentation in this study to give us a fair comparison with traditional techniques such as PCA with a small sample size. Our results suggest that AAE can tackle these challenges very well due to adversarial regularization.

To support our claim we made in Section 4.2 on the ineffectiveness of $T_{\rm KLD}^2$ statistic, we refer the reader to Figure 8, which shows the 2-D latent code of the AAE model given a set of OC samples. Since the 2-D latent code of OC samples are also close to a standard normal distribution, $T_{\rm KLD}^2$ will not be able to detect these OC behaviors. The ineffectiveness of $T_{\rm KLD}^2$ is validated in Figure 8b, where the density of $T_{\rm KLD}^2$ distribution for OC and IC samples are largely overlapped. Figure 8c shows the reconstruction error of the images given the same 2-D latent code. From these images, it is clear that the reconstructed images are very different from the OC images, and $Q_{\rm ERE}$ can be used to capture such changes. The effectiveness of $Q_{\rm ERE}$ is also shown in Figure 8d, where the distributions of IC and OC samples do not have much overlap at all.

Finally, we present qualitative results for all benchmark methods. Figure 9 shows a randomly selected sample of IC images and the reconstructions given by all the models. Apparently, deep autoencoders manage to create a better reconstruction of the IC samples with less noise. Among deep learning methods, AAE reconstructions are slightly crisper than AE and VAE. Figure 10 shows the reconstruction images of the OC samples by each model. First, we can observe that all images are not reconstructed well, which actually demonstrates the power of $Q_{\rm ERE}$ statistics for all methods.

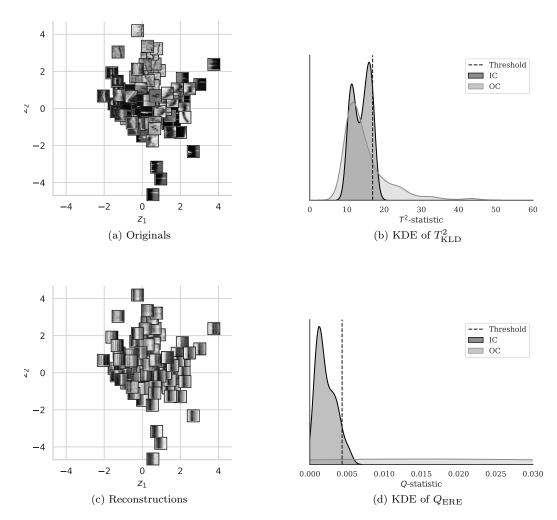


Figure 8: Mean estimations of latent code and Gaussian kernel density estimations of test statistics for an AAE model with latent dimension 2. Thresholds found from the validation set is illustrated with dashed black line.

Furthermore, we observe that deep autoencoder models, particular the regularized models VAE and AAE still generate the patterns from the IC images. However, the reconstruction images generated by the PCA methods are largely affected by the input images. For example, for class 10 and 12, the anomaly images show the horizontal patterns (i.e., abnormal patterns). All deep autoencoders reconstruct the images with IC patterns with vertical patterns (i.e., normal patterns), whereas PCA output blurry horizontal patterns. This shows PCA is unable to learn the high-level representations of IC samples and can leads to overfitting.

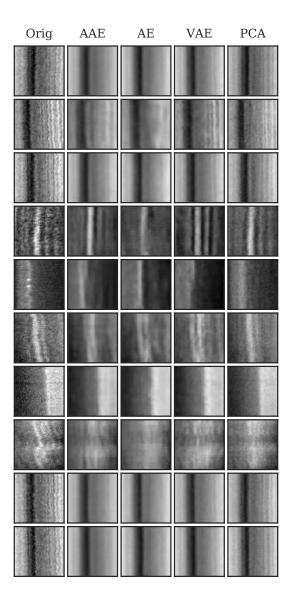


Figure 9: Example reconstructions produced by AAE, AE, VAE and PCA methods for a randomly selected IC sample, compared to the original image shown in the first column.

6 Conclusions

We proposed deep probabilistic autoencoding models to address the Phase-II analysis of high-dimensional process variables with nonlinear sources of variation. Our findings suggest that deep probabilistic autoencoding models are suitable to model the nonlinear variational patterns. We propose two process monitoring statistics, namely the $T_{\rm KLD}^2$ and $Q_{\rm ERE}$ to accurately detect OC behavior. We show that these two statistics are natural generalization of the traditional T^2 and Q

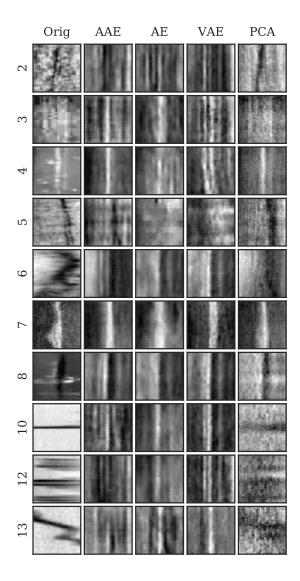


Figure 10: Example reconstructions produced by each method for each OC class along with the original version. The first column is the original image while the second to fifth columns are AAE, AE, VAE and PCA reconstructions respectively. Each row is a different OC class and the row order is the same as in Table 3.

chart for PCA.

Finally, with extensive simulation and case study comparison, we find that the monitoring statistics in the observation space (i.e., $Q_{\rm ERE}$) is typically much more effective than the monitoring statistics in the latent space (i.e., T^2), especially for deep probabilistic autoencodeing models. These findings can be further useful to guide the researchers and practitioners to design other types of

monitoring statistics focusing on the differences between the original and the reconstructed OC images. For deep autoencoders, we also encourage reconstruction error on the validation set to guide hyperparameter optimization. Finally, we find that AAE is a better model over other benchmark methods such as AE, VAE, and PCA, due to its flexible prior regularization in terms of image reconstruction accuracy and process monitoring accuracy.

References

- [1] J. Zhu and D. K. Lin, "Monitoring the slopes of linear profiles," *Quality Engineering*, vol. 22, no. 1, pp. 1–12, 2009.
- [2] C. Zou, X. Ning, and F. Tsung, "Lasso-based multivariate linear profile monitoring," *Annals of Operations Research*, vol. 192, no. 1, pp. 3–19, 2012.
- [3] J. D. Williams, W. H. Woodall, and J. B. Birch, "Statistical monitoring of nonlinear product and process quality profiles," *Quality and Reliability Engineering International*, vol. 23, no. 8, pp. 925–941, 2007.
- [4] W. A. Jensen and J. B. Birch, "Profile monitoring via nonlinear mixed models," *Journal of Quality Technology*, vol. 41, no. 1, pp. 18–34, 2009.
- [5] R. Noorossana, A. Saghaei, and A. Amiri, Statistical analysis of profile monitoring. John Wiley & Sons, 2011, vol. 865.
- [6] M. R. Maleki, A. Amiri, and P. Castagliola, "An overview on recent profile monitoring papers (2008–2018) based on conceptual classification scheme," Computers & Industrial Engineering, vol. 126, pp. 705–728, 2018.
- [7] C. Zou, F. Tsung, and Z. Wang, "Monitoring profiles based on nonparametric regression methods," *Technometrics*, vol. 50, no. 4, pp. 512–526, 2008.
- [8] P. Qiu, C. Zou, and Z. Wang, "Nonparametric profile monitoring by mixed effects modeling," *Technometrics*, vol. 52, no. 3, 2010.
- [9] C. Zou, P. Qiu, and D. Hawkins, "Nonparametric control chart for monitoring profiles using change point formulation and adaptive smoothing," *Statistica Sinica*, vol. 19, no. 3, p. 1337, 2009.
- [10] S. I. Chang and S. Yadama, "Statistical process control for monitoring non-linear profiles using wavelet filtering and b-spline approximation," *International Journal of Production Research*, vol. 48, no. 4, pp. 1049–1068, 2010.
- [11] K. Paynabar and J. Jin, "Characterization of non-linear profiles variations using mixed-effect models and wavelets," *IIE Transactions*, vol. 43, no. 4, pp. 275–290, 2011.
- [12] R. Y. Liu, "Control charts for multivariate processes," Journal of the American Statistical Association, vol. 90, no. 432, pp. 1380–1387, 1995.
- [13] K. Paynabar, P. Qiu, and C. Zou, "A change point approach for phase-i analysis in multivariate profile monitoring and diagnosis," *Technometrics*, no. just-accepted, pp. 00–00, 2015.

- [14] K. Paynabar, J. Jin, and M. Pacella, "Monitoring and diagnosis of multichannel nonlinear profile variations using uncorrelated multilinear principal component analysis," *IIE Transactions*, vol. 45, no. 11, pp. 1235–1247, 2013.
- [15] M. Grasso, B. Colosimo, and M. Pacella, "Profile monitoring via sensor fusion: the use of pca methods for multi-channel data," *International Journal of Production Research*, vol. 52, no. 20, pp. 6110–6135, 2014.
- [16] H. Yan, K. Paynabar, and J. Shi, "Image-based process monitoring using low-rank tensor decomposition," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 1, pp. 216–227, 2015.
- [17] Z. Shi, D. W. Apley, and G. C. Runger, "Discovering the nature of variation in nonlinear profile data," *Technometrics*, vol. 58, no. 3, pp. 371–382, Jul. 2016.
- [18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
- [19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [20] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.
- [21] P. Howard, D. W. Apley, and G. Runger, "Identifying nonlinear variation patterns with deep autoencoders," *IISE Transactions*, vol. 50, no. 12, pp. 1089–1103, 2018.
- [22] W. Yan, P. Guo, Z. Li et al., "Nonlinear and robust statistical process monitoring based on variant autoencoders," Chemometrics and Intelligent Laboratory Systems, vol. 158, pp. 31–40, 2016.
- [23] Z. Zhang, T. Jiang, S. Li, and Y. Yang, "Automated feature learning for nonlinear process monitoring—an approach using stacked denoising autoencoder and k-nearest neighbor rule," *Journal of Process Control*, vol. 64, pp. 49–61, 2018.
- [24] Z. Zhang, T. Jiang, C. Zhan, and Y. Yang, "Gaussian feature learning based on variational autoencoder for improving nonlinear process monitoring," *Journal of Process Control*, vol. 75, pp. 136–155, 2019.
- [25] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," arXiv preprint arXiv:1312.6114, 2013.
- [26] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," arXiv preprint arXiv:1511.05644, 2015.
- [27] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," Journal of Machine Learning Research, vol. 11, no. Dec, pp. 3371–3408, 2010.
- [28] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

- [29] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Proceedings of the 28th International Conference on Machine Learning*. Omnipress, 2011, pp. 833–840.
- [30] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," arXiv preprint arXiv:1312.6114, 2013.
- [31] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," arXiv preprint arXiv:1511.05644, 2015.
- [32] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in neural information processing systems*, 2016, pp. 2234–2242.
- [33] Q. Chen, U. Kruger, M. Meronk, and A. Leung, "Synthesis of t2 and q statistics for process monitoring," *Control Engineering Practice*, vol. 12, no. 6, pp. 745–755, 2004.
- [34] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 61, no. 3, pp. 611–622, 1999.
- [35] J. Duchi, "Derivations for linear algebra and optimization," Berkeley, California, vol. 3, 2007.
- [36] F. Locatello, S. Bauer, M. Lucic, S. Gelly, B. Schölkopf, and O. Bachem, "Challenging common assumptions in the unsupervised learning of disentangled representations," arXiv preprint arXiv:1811.12359, 2018.
- [37] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," arXiv preprint arXiv:1511.06434, 2015.
- [38] H. Abdi and L. J. Williams, "Principal component analysis," Wiley Interdisciplinary Reviews: Computational Statistics, vol. 2, no. 4, pp. 433–459, 2010.