Robust Generalized Scalar-on-Tensor Regression

Elif Konyar [a], Mostafa Reisi Gahrooei*[a] and Ruizhi Zhang [b]
[a] Department of Industrial and Systems Engineering,
University of Florida, Gainesville, FL
[b] Department of Statistics, University of Georgia, Athens, GA

Abstract

High-dimensional (HD) data, such as images and profiles, are commonly collected from complex systems and contain significant explanatory and predictive information for effective systems monitoring and control. Therefore, developing accurate and robust predictive models based on HD data is crucial. In literature, various methods, including linear scalar-on-tensor regression, are developed to model a complex system based on HD data. However, existing estimation techniques ignore the presence of outliers and are prone to biased estimations. This paper proposes a robust scalar-on-tensor regression framework that handles multi-dimensional HD input data when the data contain outliers. Our proposed estimation method is constructed using maximum Lq-likelihood estimation instead of the classical maximum likelihood estimation. The asymptotic analysis under the Gaussian distribution assumption and the guideline on choosing the tuning parameter of our proposed method is provided. Several simulations and case studies evaluate the proposed method's efficacy compared to several benchmark methods in the literature.

Keywords: Robust estimator, Scalar-on-Tensor regression, Lq-likelihood, Asset lifetime prediction.

1 Introduction

Nowadays, high-dimensional (HD) data, including profiles and images, are vastly available. For example, in condition monitoring of assets, vibration signals and thermal images are collected to predict the time to failure of an asset (Fang et al., 2019; Li et al., 2021). Accurate and robust statistical models developed based on such HD data can be used for systems monitoring, optimization, and improvement. These statistical models benefit many applications including manufacturing (Yan et al., 2019; Fang et al., 2019; Gahrooei et al., 2021), healthcare (Zhou et al., 2013), and agriculture (Ogden et al., 2002; Li et al., 2020). Specifically, regression models developed based on a sample of HD data to predict a trait value (scalar) of a system are of particular importance and are the focus of this article.

Developing predictive models based on HD data requires addressing challenges caused by the high dimensionality of data, including small sample sizes and complex correlation structures. Many authors dealt with these challenges by proposing functional and tensor regression techniques. For example, Bayesian, adaptive, and penalized function-on-scalar models have been proposed in recent years (Chen et al., 2016; Fan and Reimherr, 2017; Barber et al., 2017; Kowal and Bourgeois, 2020). Zhou et al. (2013) developed a scalar-on-tensor regression framework to predict a neurological disorder based on neuroimaging data. Li et al. (2018) extended this approach by using Tucker decomposition. Guhaniyogi et al. (2017) proposed a Bayesian version of the tensor regression with a scalar response. Fang et al. (2019) developed a penalized location-scale tensor regression model to predict the remaining useful life of a system. Also, several studies developed efficient algorithms for tensor regression estimation (Chen et al., 2019; Zhang et al., 2020). These approaches are designed based on the assumption that the data is outlier-free. Therefore, they may produce large biased estimations and predictions when the data contains outlier samples.

In most real-world applications, outlier samples exist and may create bias in model predictions if not addressed appropriately. For example, in asset management, signals such as infrared images are used for the prediction of the remaining lifetime of an asset (Fang et al., 2019). However, given the condition signals, assets with shorter or longer lifetimes may exist. Figure 1 illustrates the failure time of rotary machines versus a feature (first principal component score) of thermal images acquired by infrared cameras. As it is depicted, a few outliers exist in the data. As another example, in agriculture, UAV-captured hyperspectral images are used to predict the yield of the crop (Kanning et al., 2018). But, some plots of a crop may show abnormally higher or lower yields. Developing algorithms for HD datasets that are robust to such outliers is critical for adequate systems modeling.

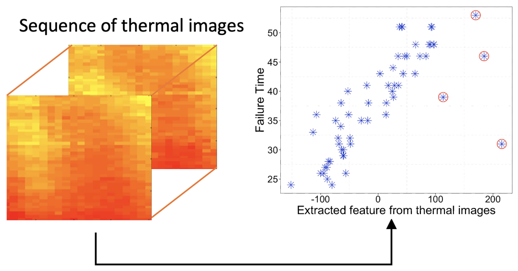


Figure 1: The lifetime of a few of the rotary systems appeared as outliers.

In the context of robust linear regression, M-estimation plays an important role in constructing robust estimators that are not sensitive to outliers. The key idea is to replace the squared loss in the ordinary least squares with some loss functions that put less weight on the outliers. There are many commonly used loss functions, e.g., Huber's loss (Huber, 1964), Tukey's Bisquare loss (Beaton and Tukey, 1974), Welsch's exponential squared loss (Dennis Jr and Welsch, 1978). In the context of high-dimensional settings where the number of parameters is larger than the sample size, regularized M-estimations are often used to obtain robust and sparse estimations. Lambert-Lacroix and Zwald (2011) proposed a robust variable selection method by using Huber's loss and adaptive lasso penalty. Li et al. (2011) show that the M-estimation with some nonconcave penalty functions can simultaneously perform parameter estimation and variable selection. The exponential squared loss combined with adaptive lasso penalty is used by Wang et al. (2013) to construct a robust estimator for sparse estimation and variable selection. Chang et al. (2018) proposed a robust estimator by combining Tukey's biweight loss with adaptive lasso penalty. When the true distributions are fully specified, Ferrari and Yang (2010) propose a robust estimation method by using Lq-likelihood. All these approaches require representing data as vectors, which break down the spatial structure of HD data points such as waveforms and images. That is, they cannot exploit the spatial correlation structure of HD data. Recently, robust functional regression approaches have been proposed (Maronna and Yohai, 2013; Wang et al., 2019; Kalogridis and Van Aelst, 2019). While these methods are suitable for modeling waveform signals, they are difficult to be extended to higher dimensions and cannot capture the between-correlation structure of multi-channel profiles.

This paper proposes a robust generalized tensor regression approach using maximum Lq-likelihood estimation to address the aforementioned challenges. Following Zhou et al. (2013), we formulate a tensor regression model with a scalar response and propose to use the Lq-likelihood to derive the robust estimation for scalar-on-tensor regression. We also show that under the Gaussian assumption, our robust estimator is a special case of M-estimator with Welsch's exponential squared loss (Dennis Jr and Welsch, 1978). To avoid overfitting due to the estimation of a large number of parameters, we use a low-rank decomposition of model parameters.

The rest of the article is organized as follows: In Section 2, we introduce notations and review some of the multilinear algebra concepts used in the article. In Section 3, we review the scalar-on-tensor regression model and illustrate the solution for estimating the parameters. In Section 4, we introduce robust scalar-on-tensor (RSoT) regression and its penalized form. We discuss the choices of tuning parameters for our proposed method in Section 4.3. Section 5 shows the asymptotic properties of the proposed robust estimator under the Gaussian distribution assumption. In Section 6, three simulation studies are conducted to compare the performance of the RSoT against benchmark methods in terms of prediction errors. Section 7 introduces the case study of estimating the failure time of an engine given thermal images. In Section 8, we provide our discussion on the proposed framework. Finally, Section 9 concludes the paper and provides some insights regarding future work.

2 Tensor Notation and Multilinear Algebra

In this section, we introduce the notations and basic tensor algebra used in this paper. Throughout the paper, we denote a scalar by a lower or upper case letter, e.g., a or A; a vector by a boldface lowercase letter and a matrix by a boldface uppercase letter, e.g., $\boldsymbol{a}$ and $\mathbf{A}$; and a tensor by a calligraphic letter, e.g., $\mathcal{A}$. For example, we denote an order-n tensor by $\mathcal{R} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_n}$, where I_i is the dimension of the i^{th} mode of tensor $\mathcal{R}$. We also denote a mode-j matricization of tensor $\mathcal{R}$ as $\boldsymbol{R}_{(j)} \in \mathbb{R}^{I_j \times I_{-j}}$, whose columns are the mode-j fibers of the corresponding tensor $\mathcal{R}$, and $I_{-j} = I_1 \times I_2 \times \cdots \times I_{j-1} \times I_{j+1} \times \cdots \times I_n$. The $vec(\mathcal{R})$ operator transforms the entries of a n-dimensional tensor $\mathcal{R} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_n}$ into a column vector. Furthermore, the Kronecker product of two matrices $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{B} \in \mathbb{R}^{r \times s}$ is denoted as $\boldsymbol{A} \otimes \boldsymbol{B} \in \mathbb{R}^{mr \times ns}$ and is obtained by multiplying each element of matrix $\boldsymbol{A}$ to the entire matrix $\boldsymbol{B}$:

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B} & \dots & a_{1n}\boldsymbol{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\boldsymbol{B} & \dots & a_{mn}\boldsymbol{B} \end{bmatrix}.$$

If $\boldsymbol{A}$ and $\boldsymbol{B}$ have the same number of columns, i.e., $n = s$, then the Khatri-Rao product is defined as the mr-by-n columnwise Kronecker product, $\boldsymbol{A} \odot \boldsymbol{B} = [\boldsymbol{a}_1 \otimes \boldsymbol{b}_2 \boldsymbol{a}_2 \otimes \boldsymbol{b}_2 \cdots \boldsymbol{a}_n \otimes \boldsymbol{b}_n]$. Finally, the outer product of n vectors $\boldsymbol{a}_i \in \mathbb{R}^{I_i}, i = 1, 2, \cdots, n$, is denoted by $\boldsymbol{a}_1 \circ \boldsymbol{a}_2 \cdots \circ \boldsymbol{a}_n$

and is a $I_1 \times I_2 \times \cdots \times I_n$ tensor with entries $(\boldsymbol{a}_1 \circ \boldsymbol{a}_2 \cdots \circ \boldsymbol{a}_n)_{i_1,i_2,\cdots,i_n} = \prod_{k=1}^{n} a_{k,i_k}$. We denote the trace of a matrix $\boldsymbol{A}$ by $\mathrm{Tr}(\boldsymbol{A})$.

3 Review of Generalized Scalar-on-Tensor Regression

In this section, we review the generalized scalar-on-tensor (SoT) framework proposed by Zhou et al. (2013). Assume a set of training data of size N that includes response variables $y_i \in \mathbb{R}; (i = 1, \cdots, N)$ and input tensors $\mathcal{X}_i \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_m}; (i = 1, \cdots, N)$ is available. Assume y_i follows an exponential family distribution with canonical parameters θ_i and dispersion parameter ϕ,

$$y_i | \mathcal{X}_i \sim f(y_i; \theta_i, \phi) = \exp\left\{ \frac{y_i \theta_i - b(\theta_i)}{a(\phi)} + c(y_i, \phi) \right\}, \tag{1}$$

where $b(.)$, $a(.)$ and $c(.)$ are distribution-specific known functions. Then, SoT regression characterizes a generalized linear tensor model (GLM) as follows:

$$g(E[y_i|\mathcal{X}_i]) = \mu + \langle \mathcal{X}_i, \mathcal{B}_0 \rangle + e_i;\ i = 1, 2, \cdots, N. \tag{2}$$

Here, $\mathcal{B}_0 \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_m}$ is the true tensor of parameters to be estimated, and e_i is independent and identically distributed random noise with zero mean and constant variance. If we vectorize the tensors $\mathcal{X}_i$ and $\mathcal{B}_0$ in (2), then, the parameters $\mathcal{B}_0$ can be estimated by available vector-based methods. For instance, for Gaussian distribution, we can use the method of ordinary least squares estimation by minimizing the mean square loss function, $L = \frac{1}{N} \sum_{i=1}^{N} \left(y_i - \mu - \langle \mathcal{X}_i, \mathcal{B} \rangle\right)^2$. Nevertheless, without imposing a penalty on the parameters, this approach results in a severe over-fitting due to a large number of parameters. To overcome the high dimensionality of the parameters and the potential over-fitting problem, Zhou et al. (2013) imposed low-rankness on the tensor of parameters by including a rank-R decomposition of $\mathcal{B}$. Then, the generalized scalar-on-tensor regression model is solved by minimizing the negative log-likelihood (maximizing the log-likelihood) of the training data with low-rankness constraint as follows:

$$\min_{\mu, \mathcal{B}} \ -l(\mu, \mathcal{B}) = \sum_{i=1}^{N} -\frac{y_i \theta_i - b(\theta_i)}{a(\phi)} - \sum_{i=1}^{N} c(y_i, \phi) \tag{3}$$

$$s.t.\ \ \mathcal{B} = \sum_{r=1}^{R} \boldsymbol{u}_1^{(r)} \circ \boldsymbol{u}_2^{(r)} \cdots \circ \boldsymbol{u}_m^{(r)}, \tag{4}$$

where $\theta_i = \mu + \langle \mathcal{B}, \mathcal{X}_i \rangle$ and $\boldsymbol{u}_d^{(r)} \in \mathbb{R}^{I_d}, d = 1, 2, \cdots, m;\ r = 1, 2, \cdots, R$. For convenience, the decomposition can also be represented by a shorthand, $\mathcal{B} = [[\boldsymbol{U}_1, \boldsymbol{U}_2, \cdots, \boldsymbol{U}_m]]$, where $\boldsymbol{U}_d = [\boldsymbol{u}_d^{(1)}, \boldsymbol{u}_d^{(2)}, \cdots, \boldsymbol{u}_d^{(R)}] \in \mathbb{R}^{I_d \times R}, d = 1, 2, \cdots, m$. In this way, the number of parameters will be $1 + R(\sum_{d=1}^{m} I_d)$, which is much smaller than the number of $1 + \prod_{d=1}^{m} I_d$ parameters without the low-rankness assumption. To solve this problem, one can use the Alternating Least Squares (ALS) procedure. That is, we solve the optimization problem in (3) and estimate $\boldsymbol{u}_d^{(j)}$ by fixing the remaining parameters. Next, we repeat this procedure for the remaining parameters in a similar way. This procedure is repeated until convergence. The details of the parameter estimation are provided in Section 1 of Supplementary Materials.

4 Robust Generalized Scalar-on-Tensor Regression

In this section, we consider the problem of robust generalized scalar-on-tensor (RSoT) regression when the training data $\{(y_i, \mathcal{X}_i)\}_{i=1}^{N}$ contains some outliers. Specifically, we assume the data follows the generalized linear model with gross outlier corruption. That is, y follows a mixture distribution with probability density function $h(y; \theta, q) = (1-q) f_\theta(y) + q g(y)$, where $q \in [0, 1)$ is the proportion of the outliers, $f_\theta(y)$ is the pdf of an exponential family with canonical parameters θ, and $g(y)$ denotes the probability density function of the outlier that could be arbitrary or depend on $\mathcal{X}$. Then, we write the RSoT regression model as follows:

$$
\begin{aligned}
p(y_i; \theta_i, q) &= h_{\theta_i, q}(y_i) = (1-q) f_{\theta_i}(y_i) + q g(y_i),\ \forall i && (5)\\
f_{\theta_i}(y) &= d(y) \exp\{y\theta_i - b(\theta_i)\} && (6)\\
b'(\theta_i) &= E[y|\mathcal{X}_i] && (7)
\end{aligned}
$$

where $\theta_i = \mu + \langle \mathcal{X}_i, \mathcal{B}_0 \rangle$, $\mathcal{B}_0 = [[\boldsymbol{U}_{01}, \boldsymbol{U}_{02}, \cdots, \boldsymbol{U}_{0m}]]$, and $b(.)$ is a known function and depends on the underlying exponential family distribution, $b'(.)$ is its first derivative, and $\mathcal{B}_0$ is the true tensor of parameters, which is decomposable by a set of true factor matrices $\boldsymbol{U}_{01}, \cdots, \boldsymbol{U}_{0m}$. Note that if $f_{\theta_i}(y)$ is a single-parameter distribution from exponential family, then $\theta_i = \mu + \langle \mathcal{X}_i, \mathcal{B}_0 \rangle$. However, for two-parameter exponential family distributions, θ_i becomes a function of the parameters. For instance, when estimating mean of a Gaussian distribution with unknown variance, σ^2, then θ_i becomes $\theta_i = \frac{\gamma_i}{\sigma^2}$, where $\gamma_i = \mu + \langle \mathcal{X}_i, \mathcal{B}_0 \rangle$.

Under such gross error model when data contain outliers, the maximum likelihood loss in (3) will not accurately estimate the true parameter $\mathcal{B}_0$. Thus, to reduce the effect of outliers, in Section 4.1, we will construct a robust generalized scalar-on-tensor (RSoT) estimation procedure using maximum Lq-likelihood estimation (Ferrari and Yang, 2010). Next, we will introduce the penalized version of the Lq-likelihood to encourage sparsity in the solutions in Section 4.2. Finally, we discuss the choices of tuning parameters in our proposed method in Section 4.3.

4.1 Robust Estimation via L*q*-likelihood

We propose to estimate $\mathcal{B}_0$ and μ by solving the following optimization problem with the low-rank constraint on the tensor of parameters:

$$\min_{\mu,\mathcal{B}} \quad \delta_\alpha(\theta_i, y_i) = \sum_{i=1}^{N} \frac{1-(f_{\theta_i}(y_i))^\alpha}{\alpha} \tag{8}$$
$$s.t. \quad \mathcal{B} = \sum_{r=1}^{R} \boldsymbol{u}_1^{(r)} \circ \boldsymbol{u}_2^{(r)} \cdots \circ \boldsymbol{u}_m^{(r)},$$

where $f_{\theta_i}(y) = a(y)\exp\{y\theta_i - b(\theta_i)\}$. In this formulation, the tuning parameter, α, controls the trade-off between the statistical efficiency and the robustness of our proposed method. Note that when $\alpha \to 0$, $\delta_\alpha(\theta_i, y_i) \to \sum_{i=1}^{N} -\log(f_{\theta_i}(y_i))$. That is, the proposed Lq-likelihood becomes similar to the negative log-likelihood as in (3). On the other hand, when $\alpha \to \infty$, $\delta_\alpha(\theta_i, y_i) \to 0$. Thus, in this case, the resulting estimator may not be affected by any outliers but will lose all statistical efficiency to estimate the true parameters. Therefore, it is critical to choose the tuning parameter α for our robust estimation method. We explain the selection of α in Section 4.3.

Suppose $f_{\theta_i}(y)$ is a single-parameter exponential family distribution. Then, Problem (8) is equivalent to

$$\min_{\mu,\mathcal{B}} \sum_{i=1}^{N} \frac{1-\Big(d(y_i)\exp\{y_i(\mu + \langle \mathcal{X}_i, \mathcal{B}\rangle) - b(\mu + \langle \mathcal{X}_i, \mathcal{B}\rangle)\}\Big)^\alpha}{\alpha} \tag{9}$$
$$s.t. \; \mathcal{B} = \sum_{r=1}^{R} \boldsymbol{u}_1^{(r)} \circ \boldsymbol{u}_2^{(r)} \cdots \circ \boldsymbol{u}_m^{(r)}.$$

The estimation of the tensor $\mathcal{B}$ is now equivalent to the estimation of the matrices $\boldsymbol{U}_1, \boldsymbol{U}_2, \cdots, \boldsymbol{U}_m$. To solve the optimization problem in (9) and estimate $\boldsymbol{u}_d^{(j)}(d =$

$1, \cdots, m; j = 1, \cdots, R$), we propose to employ Newton's search method and alternating least squares (ALS) algorithms. Note that any first/second-order search methods can be used together with the ALS approach. However, Newton's search method is advantageous due to its fast convergence. First, the initial values of $\{\mu, \boldsymbol{U}_1, \boldsymbol{U}_2, \cdots, \boldsymbol{U}_m\}$ are set to the solution of SoT as it is discussed in Section 3. Then, we re-write the inner product as $\langle \mathcal{X}_i, \mathcal{B} \rangle = \mathrm{Tr}\left(\mathbf{U}_d^\top \mathcal{X}_{i,(d)} \mathbf{K}_{U_d}\right)$, where $\mathbf{K}_{U_d} = \mathbf{U}_m \odot \ldots \mathbf{U}_{d+1} \odot \mathbf{U}_{d-1} \ldots \odot \mathbf{U}_1$. Furthermore, we set $\mathbf{W}_{i,d} = [\mathbf{w}_{i,d}^{(1)}, \ldots, \mathbf{w}_{i,d}^{(R)}] := \mathcal{X}_{i,(d)} \mathbf{K}_{U_d}$ ($\mathbf{w}_{i,d}^{(j)} \in \mathbb{R}^{I_d}$), $T_{d,i} := \mathrm{Tr}\left(\mathbf{U}_d^\top \mathbf{W}_{i,d}\right)$, and $T_{d,i}^{(-j)} := \mathrm{Tr}\left(\mathbf{U}_d^{(-j)^\top} \mathbf{W}_{i,d}^{(-j)}\right)$, where j^{th} columns of $\mathbf{U}_d$ and $\mathbf{W}_{i,d}$ are removed. Then, for fixed μ, $\{\mathbf{U}_{d'}\}_{d'=1,d'\neq d}^m$, and $\{\mathbf{u}_d^{(j')}\}_{j'=1,j'\neq j}^R$, we propose to use the Newton's search algorithm to estimate $\boldsymbol{u}_d^{(j)}$ by solving,

$$\min_{\mathbf{u}_d^{(j)}} \sum_{i=1}^N \frac{1 - \left(d(y_i) \exp\{y_i(\mu + T_{d,i}^{(-j)} + \mathbf{u}_d^{(j)^\top} \mathbf{w}_{i,d}^{(j)}) - b(\mu + T_{d,i}^{(-j)} + \mathbf{u}_d^{(j)^\top} \mathbf{w}_{i,d}^{(j)})\}\right)^\alpha}{\alpha}. \tag{10}$$

We iteratively update $\mathbf{u}_d^{(j)}$ until it converges to a stationary point through Newton's search algorithm as follows:

$$\mathbf{u}_d^{(j)^{(t)}} = \mathbf{u}_d^{(j)^{(t-1)}} - \zeta_t H\left(\mathbf{u}_d^{(j)^{(t-1)}}\right)^{-1} G\left(\mathbf{u}_d^{(j)^{(t-1)}}\right),$$

where $G(.)$ is the gradient, $H(.)$ is the Hessian matrix, $\mathbf{u}_d^{(j)^{(t)}}$ is the $\mathbf{u}_d^{(j)}$ at t^{th} iteration, and ζ_t is the learning rate. The gradient and the Hessian matrix can be found in Section 2 of Supplementary Materials. The convergence criteria are either $||\mathbf{u}_d^{(j)^{(t)}} - \mathbf{u}_d^{(j)^{(t-1)}}||_2$ becomes less than a threshold, or the maximum number of iterations is reached. We select the learning rate by a backtracking algorithm.

We repeat this procedure for the $\mathbf{u}_d^{(j)}$ of the remaining modes and factors ($d = 1, \cdots, m;\ j = 1, \cdots, R$), until convergence. After learning every vector $\mathbf{u}_d^{(j)}$ ($d = 1, \cdots, m; j = 1, \cdots, R$), the intercept μ is estimated by using Newton's search algorithm by following a similar idea with $\mathbf{u}_{d,t}^{(r)}$. The details of the update procedure are provided in Section 2 of Supplementary Materials.

Our proposed method has a special structure for Gaussian distribution. In this special case, our robust estimator will reduce to

$$\min_{\mu,\mathcal{B}} \sum_{i=1}^{N} \rho_\alpha \left(y_i - \mu - \langle \mathcal{X}_i, \mathcal{B} \rangle \right) \tag{11}$$
$$s.t.\ \mathcal{B} = \sum_{r=1}^{R} \boldsymbol{u}_1^{(r)} \circ \boldsymbol{u}_2^{(r)} \cdots \circ \boldsymbol{u}_m^{(r)},$$

where $\rho_\alpha(t) = 1 - \exp(-\alpha t^2/2)$ known as the Welsch's exponential loss function (Wang et al., 2013). Note that the interpretation of the tuning parameter α in our method is the same for Welsch's exponential loss function. The details about the special case are provided in Section 3 of Supplementary Materials.

4.2 Robust Sparse Estimation through Adaptive Lasso

It often appears that many elements in the input tensor $\mathcal{X}$ contain no information in predicting the output y. Therefore, it is important to identify the informative elements of the input tensor. For this purpose, we define the penalized Lq-likelihood, which imposes a sparsity-inducing penalty on the tensor of parameters $\mathcal{B}$. This sparsity is achieved equivalently by penalizing the columns of factor matrices $\boldsymbol{U}_d$, namely $\boldsymbol{u}_d^{(r)}$, in the objective function as follows:

$$\min_{\mu,\mathcal{B}} \sum_{i=1}^{N} \frac{1 - \Big(d(y_i) \exp\{ y_i(\mu + \langle \mathcal{X}_i, \mathcal{B} \rangle) - b(\mu + \langle \mathcal{X}_i, \mathcal{B} \rangle) \} \Big)^{\alpha}}{\alpha} + \frac{N}{mR} \sum_{d=1}^{m} \sum_{r=1}^{R} \| \boldsymbol{c}_d^{(r)} * \boldsymbol{u}_d^{(r)} \|_1 \tag{12}$$

$$s.t.\ \ \mathcal{B} = \sum_{r=1}^{R} \boldsymbol{u}_1^{(r)} \circ \boldsymbol{u}_2^{(r)} \cdots \circ \boldsymbol{u}_m^{(r)}.$$

Here, $\boldsymbol{c}_d^{(r)} \in \mathbb{R}^{I_d}$ is a vector of weights and $\boldsymbol{c} * \boldsymbol{u}$ is an element-wise multiplication of vectors. The sparsity-inducing penalty in (12) resembles the adaptive lasso. The solution to this problem is an extension of the RSoT algorithm and gives sparse estimation of model parameters. Assuming μ, $\boldsymbol{U}_m, \cdots, \boldsymbol{U}_{d+1}, \boldsymbol{U}_{d-1} \cdots, \boldsymbol{U}_1$ and $\{\mathbf{u}_d^{(j')}\}_{j'=1, j' \neq j}^{R}$ are given, we use an alternating procedure to estimate $\mathbf{u}_d^{(j)}$ by solving the problem as follows:

$$\min_{\mathbf{u}_d^{(j)}} \sum_{i=1}^{N} \frac{1 - \Big(d(y_i) \exp\{ y_i(l_{d,i}^{j} + \mathbf{u}_d^{(j)\top} \mathbf{w}_{i,d}^{(j)}) - b(l_{d,i}^{j} + \mathbf{u}_d^{(j)\top} \mathbf{w}_{i,d}^{(j)}) \} \Big)^{\alpha}}{\alpha} + \frac{N}{mR} \| \boldsymbol{c}_d^{(j)} * \boldsymbol{u}_d^{(j)} \|_1, \tag{13}$$

where $l^j_{d,i} = \mu + T^{(-j)}_{d,i}$. We discuss the choice of the penalization parameter $\boldsymbol{c}^{(j)}_d$ in Section 4.2. We employ the proximal Newton algorithm to solve the optimization problem in (13). Note that other alternative methods, such as proximal gradient descent, can also be applied to solve the adaptive lasso problem. The proximal function is the solution to the following adaptive lasso problem:

$$prox_\eta(\boldsymbol{u}) = \boldsymbol{z}^* = \arg\min_{\boldsymbol{z}} \frac{1}{2\eta}\|\boldsymbol{u} - \boldsymbol{z}\|_2^2 + \frac{N}{mR}\|\boldsymbol{c} * \boldsymbol{z}\|_1.$$

The update of vector $\mathbf{u}^{(j)}_d$ with learning rate ζ is then obtained as $\boldsymbol{u}^{(j)}_d = prox_\eta(\boldsymbol{u}^{(j)}_d - \zeta H(\mathbf{u}^{(j)}_d)^{-1} G(\mathbf{u}^{(j)}_d))$.

4.3 Selection of Tuning Parameters

The proposed method requires tuning a few hyperparameters, including the robustness parameter, α, the model rank, R, and the regularization parameter, $\mathbf{c}^{(j)}_d$. In the proposed method, α is a critical tuning parameter, as it controls the robustness and the quality of the fit of our proposed method. We perform a joint parameter selection mechanism to identify these hyperparameters in RSoT. That is, we perform a five-fold cross-validation and select (α, R) pair, which minimizes a robust performance measure such as truncated Root Mean Squared Error (tRMSE). When performing RSoT with the adaptive lasso penalty, we select the three parameters jointly. More specifically, given (α, R), we propose to minimize a BIC-type objective function to select the regularization parameter $\boldsymbol{c}^{(j)}_d$:

$$\min_{\boldsymbol{c}^{(j)}_d \in \mathbb{R}^{I_d}} \left(\sum_i \delta_\alpha(\theta_i, y_i) + N \sum_{k=1}^{I_d} c^{(j)}_{d,k} \left| u^{(j)(RSoT)}_{d,k} \right| - \sum_{k=1}^{I_d} \log(0.5 N c^{(j)}_{d,k}) \log(N) \right), \tag{14}$$

which yields $\hat{c}^{(j)}_{d,k} = \frac{\log(N)}{N\left|u^{(j)(RSoT)}_{d,k}\right|}$.

In this selection mechanism, as the first step, we train the $RSoT$ model for different (α, R) pairs and obtain the corresponding $u^{(j)(RSoT)}_d$ $(d = 1, \ldots, M;\ j = 1, \ldots, R)$ for each pair. Next, $\boldsymbol{c}^{(j)}_d$ is computed given the estimated $u^{(j)(RSoT)}_d$. As the final step, the $\left(\alpha, R, \boldsymbol{c}^{(j)}_d\right)$ tuple is selected based on five-fold cross-validation with tRMSE as the performance metric.

To determine the range of α, we experimentally search for a large enough value of α which causes the estimator to lose its statistical power (too robust and ignores all the data) and results in unacceptable prediction performance. Next, we determine the α set by starting with that large value and logarithmically decreasing it. The user should determine the number of values in the range based on the available computational resources. Alternatively, one can reduce the value of α as long as the prediction performance improves. The specific sets used in the simulations and case studies are provided in the corresponding sections.

5 Asymptotic Analysis Under Gaussian Assumption

In this section, we study the statistical properties of our proposed robust estimator under normal distribution assumption, i.e.,

$$\begin{aligned} y &= \mu + \langle \mathcal{X}, \mathcal{B}_0 \rangle + \epsilon, \\ \epsilon &\sim N(0, \sigma^2), \end{aligned} \tag{15}$$

which yields to exponential loss defined by $\rho_\alpha(t) = 1 - \exp(-\alpha t^2/2)$. For simplicity, we omit the intercept μ, though similar conclusions still hold with the intercept term. To study the asymptotic properties, we follow the classical asymptotic setting when the dimension of the tensor $\mathcal{B}$ is fixed and the sample size N can go to ∞.

Define $\psi_\alpha(t) = \rho'_\alpha(t)$ be the derivative of the exponential squared loss. First, the following lemma characterizes the gradient and Hessian of the loss $\rho_\alpha\left(y - \langle \mathcal{X}, \mathcal{B} \rangle\right)$.

Lemma 1. *For a rank-R decomposition of a tensor* $\mathcal{B} = [[\boldsymbol{U}_1, \boldsymbol{U}_2, \cdots, \boldsymbol{U}_m]]$, *we have*

$$\begin{aligned} \nabla(\rho_\alpha\left(y - \langle \mathcal{X}, \mathcal{B} \rangle\right)) &= -\,\psi_\alpha(y - \langle \mathcal{X}, \mathcal{B} \rangle)\boldsymbol{J}^\top(vec(\mathcal{X})), \\ \nabla^2(\rho_\alpha\left(y - \langle \mathcal{X}, \mathcal{B} \rangle\right)) &= -\,\psi_\alpha(y - \langle \mathcal{X}, \mathcal{B} \rangle)\nabla(\boldsymbol{J}^\top vec(\mathcal{X})) \\ &\quad + \psi'_\alpha(y - \langle \mathcal{X}, \mathcal{B} \rangle)\boldsymbol{J}^\top(vec(\mathcal{X}))(vec(\mathcal{X}))^\top \boldsymbol{J}, \end{aligned}$$

where $\boldsymbol{J}_i = \boldsymbol{\Pi}_i\left[(\boldsymbol{U}_m \odot \cdots \odot \boldsymbol{U}_{i+1} \odot \boldsymbol{U}_{i-1} \cdots \odot \boldsymbol{U}_1) \otimes I_{I_i}\right] \in \mathbb{R}^{(\prod_{j=1}^m I_j) \times (I_i R)}$, $\boldsymbol{\Pi}_i$ *is the* $(\prod_{j=1}^m I_j) \times (\prod_{j=1}^m I_j)$ *permutation matrix that reorders* $vec(\mathcal{B}_{(i)})$ *to obtain* $vec(\mathcal{B})$, *i.e.,* $vec(\mathcal{B}) = \boldsymbol{\Pi}_i vec(\mathcal{B}_{(i)})$, *and* $\boldsymbol{J} = [\boldsymbol{J}_1, \boldsymbol{J}_2, \cdots, \boldsymbol{J}_m]$.

The proof of Lemma 1 is provided in Section 4 of the Supplementary Materials. Next, we discuss the identifiability issue. Since the rank-R decomposition $\mathcal{B} = [[\boldsymbol{U}_1, \boldsymbol{U}_2, \cdots, \boldsymbol{U}_m]]$ is indeterminable due to scaling and permutation, we borrow the idea in Zhou et al. (2013)

and consider the following parameter space:

$$\mathcal{U} = \{(\boldsymbol{U}_1, \cdots, \boldsymbol{U}_m : u_{d,1}^{(j)} = 1 \text{for} d = 1, 2, \cdots, m-1, j = 1, 2, \cdots, R, \text{and} \\ , u_{m,1}^{(1)} > u_{m,1}^{(2)} > \cdots > u_{m,1}^{(R)})\},$$

which is open and convex. In the set $\mathcal{U}$, the factor matrices $U_1, \cdots, U_{m-1}$ are scaled in a way that their first row is ones, which determines the first row of U_m that is permuted in a decreasing order to solve the permutation issue. Then, the tensor $\mathcal{B}$ with the rank-R decomposition in the set $\mathcal{U}$ is unique to scaling and permutation. Now, we are ready to present the asymptotic distribution of our robust estimator.

Theorem 1. *Assume $(y_i, \mathcal{X}_i)_{i=1,2,\cdots,N}$ are i.i.d. following the scalar-on-tensor regression model with Gaussian assumption in (15). If the true parameter $\mathcal{B}_0 = [[\boldsymbol{U}_{01}, \cdots, \boldsymbol{U}_{0m}]] \in \mathcal{U}$, then our proposed RSoT estimator $\hat{\mathcal{B}}$ is consistent up to a permutation. That is, $\hat{\mathcal{B}}$ converges in probability to $\mathcal{B}_0$ up to a permutation.*

Theorem 2. *Assume $(y_i, \mathcal{X}_i)_{i=1,2,\cdots,N}$ are i.i.d. following the scalar-on-tensor regression model in (5) and the residual term ϵ_i follows a symmetric distribution with mean* 0. *Define*

$$V(\boldsymbol{U}_1, \cdots, \boldsymbol{U}_m) = \boldsymbol{J}^\top \Big[\sum_{i=1}^{N} (vec(\mathcal{X}_i))(vec(\mathcal{X}_i))^\top \Big] \boldsymbol{J}, \tag{16}$$

where $\boldsymbol{J} = [\boldsymbol{J}_1, \boldsymbol{J}_2, \cdots, \boldsymbol{J}_m]$. Assume the matrix $V(\boldsymbol{U}_{01}, \cdots, \boldsymbol{U}_{0m})$ is nonsingular for the true parameter $\mathcal{B}_0 = [[\boldsymbol{U}_{01}, \cdots, \boldsymbol{U}_{0m}]] \in \mathcal{U}$. Then,

$$\sqrt{n} \left(vec(\hat{\mathcal{B}}) - vec(\mathcal{B}_0) \right) \to N(0, \frac{\mathbf{E}\,(\psi_\alpha(\epsilon))^2}{[\mathbf{E}\,(\psi'_\alpha(\epsilon))]^2} \, (V(\boldsymbol{U}_{01}, \cdots, \boldsymbol{U}_{0m}))^{-1}). \tag{17}$$

The proofs of Theorems 1 and 2 are provided in Section 4 of the Supplementary Materials. We should emphasize that the asymptotic results in Theorem 1 and Theorem 2 imply that, under the normal assumption, the optimal solution of the optimization problem in (8) will be close to the true parameter when no outliers exist. Although these theoretical results do not imply the robustness of the proposed method, we conduct extensive simulation studies and case studies in the next sections to illustrate the robustness and efficiency of our proposed method. In addition, these theorems do not show the convergence of our algorithm to the global optimal solution. Nevertheless, our empirical analysis (presented in the next section) shows that the proposed algorithm converges to a stationary point in all simulations (please see examples in Section 7 of Supplementary Materials).

6 Performance Evaluation Using Simulation

In this section, we conduct simulation studies to evaluate the performance of the proposed method. We seek to compare the performance of the RSoT with the existing methods under different distribution assumptions. Therefore, we consider three simulation scenarios. In the first simulation, we assume the target follows Gaussian distribution. In the second simulation, the target is assumed to follow the Poisson distribution. In these two simulation settings, we do not consider sparsity. In the third simulation, we assume the target follows a normal distribution and that the model parameters are sparse. In our simulations, we compare the prediction performance of the RSoT with three benchmarks, namely SoT proposed by Zhou et al. (2013), penalized exponential squared loss (ESL-LASSO) proposed by Wang et al. (2013) and robust linear regression (RLR) with Huber's loss. To perform ESL-LASSO and RLR, we vectorize the input tensor. ESL-LASSO method handles the high-dimensionality of the data by performing variable selection using a lasso penalty. The method has a hyperparameter that determines its level of robustness. The proposed data-driven method in Wang et al. (2013), which is based on minimizing the asymptotic variance of the parameter estimates, does not work properly in the high-dimensional case, as the asymptotic covariance matrix becomes singular. Thus, we perform cross-validation (CV) to tune this benchmark's hyperparameter. Furthermore, RLR cannot handle the situation where the sample size n is larger than the number of features p (i.e., $p >> n$). Therefore, we perform Principal Component Analysis (PCA) to reduce the dimensionality and extract low dimensional features of the input data. Then, we apply RLR on the target given the extracted scores by PCA. We evaluate the performance of RLR with PCA, denoted as PCA-RLR, and without PCA, denoted as RLR.

6.1 Simulation I: Gaussian Distribution

We first randomly generate a set of N training data $\{(y_i, \mathcal{X}_i)\}$ containing input tensors $\mathcal{X}_i \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and output values y_i $(i = 1, \cdots, N)$ as follows. Let $\boldsymbol{x}_i \in \mathbb{R}^{I_1 I_2 I_3}$ be a random vector that follows a multivariate normal distribution with mean $\boldsymbol{\mu}_x$, and the covariance matrix $\boldsymbol{\Sigma}$, i.e., $\boldsymbol{x}_i \sim \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma})$. We generate the $(i, j)^{th}$ entry of $\boldsymbol{\Sigma}$ as $\rho^{|i-j|}$. Tensor $\mathcal{X}_i$ is then constructed by reshaping vector $\boldsymbol{x}_i$. The factor matrices $\boldsymbol{U}_i \in \mathbb{R}^{I_i \times R}, (i = 1, 2, 3)$ are simulated elementwise from a normal distribution with mean zero and variance σ_u^2. That is, each entry of $\boldsymbol{U}_i$ follows $N(0, \sigma_u^2)$. The tensor of parameters is then constructed as $\mathcal{B} = \sum_{r=1}^{R} \boldsymbol{u}_1^{(r)} \circ \boldsymbol{u}_2^{(r)} \circ \boldsymbol{u}_3^{(r)}$. Furthermore, we simulate responses by $y_i = \langle \mathcal{B}, \mathcal{X}_i \rangle + e_1$ where e_1 is a random error simulated from a normal distribution with mean zero and variance σ_1^2. Then, we randomly select q % of the instances and add a random error, e_2, simulated from a normal distribution with mean ν and variance $\sigma_2^2 > \sigma_1^2$, i.e., $y_i = \langle \mathcal{B}, \mathcal{X}_i \rangle + e_1 + e_2$. These instances are considered outliers.

When generating data, we set $\boldsymbol{\mu}_x = \mathbf{0.1} \in \mathbb{R}^{I_1 I_2 I_3}$, $\sigma_u = 0.5$, $\rho = 0.001$, $\sigma_1 = 0.5$, $\sigma_2 = 1$, $I_1 = 10, I_2 = 10, I_3 = 10$, and $R = 2$. We simulate training data sets of size $N = 315$ by choosing the mean of outlier distribution $\nu \in \{1, 2, 3, 4, 5\}$ and $q \in \{0\%, 3\%, 5\%, 8\%, 10\%, 15\%\}$. Note that $q = 0\%$ means no outlier is added to training data. Using the simulated data, we train a model for each method (i.e., RSoT, SoT, ESL-LASSO, RLR). For the RSoT method, we select the rank from the set $\{2, 3, 4\}$, and α from the set $\{2, 1.414, 1, 0.707, 0.5, 0.353\}$ (i.e., $\frac{4}{2^{k/2}}$, $k = 2, \dots, 7$) by using cross-validation as explained in Section 4.3. Our cross-validation performance measure is truncated RMSE, calculated based on 90 % of the instances of the least squared errors. Next, we generate a set of 135 test data to evaluate the performance of each method. The testing dataset is generated as explained before, except that it does not contain outliers. We calculate the root mean square prediction error (RMSPE) as the performance measure, computed as $\text{RMSPE} = \sqrt{\frac{\sum_{i=1}^{N} (y_i - \widehat{y}_i)^2}{N}}$, where y is the actual target, and $\widehat{y}$ is the predicted value. This procedure is repeated 25 times to capture the variance of the RMSPE. In each scenario, we compare the proposed method with benchmarks based on the RMSPE calculated at different outlier magnitudes (ν) and the percentage of outliers (q). Table 1 reports the average and standard deviation of RMSPE in each scenario.

As it is reported in Table 1, under all the outlier magnitudes and the percentage of outliers, the RSoT outperforms the benchmarks. For example, when $\nu = 3$ and $q = 10\%$, the RMSPEs are 0.611, 0.850, 11.330, and 24.671 for the RSoT, SoT, ESL-LASSO, and RLR, respectively. In addition, the performance of the RSoT is more stable compared to the benchmarks when both the outlier magnitude and the percentage of outliers increase. Furthermore, when no outlier exists, RSoT outperforms ESL-LASSO and RLR and achieves similar performance to SoT. Thus, it is still advantageous to use RSoT when it is unknown

Table 1: Simulation I (Gaussian Case): Comparison between the proposed method (RSoT) and the benchmarks in terms of RMSPE.

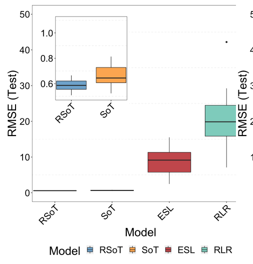
q (%)	ν	RSoT	SoT	ESL-LASSO	RLR
0	0	0.581 (0.047)	**0.574** (0.044)	10.825 (4.679)	21.711 (7.174)
3	1	**0.581** (0.034)	0.587 (0.037)	10.667 (4.526)	22.910 (7.149)
3	2	**0.593** (0.056)	0.625 (0.051)	13.936 (11.832)	23.624 (7.800)
3	3	**0.584** (0.046)	0.664 (0.080)	8.550 (3.757)	20.244 (7.368)
3	4	**0.602** (0.046)	0.730 (0.089)	8.968 (3.078)	24.615 (11.659)
3	5	**0.581** (0.042)	0.788 (0.114)	11.747 (7.406)	22.412 (7.177)
5	1	**0.603** (0.050)	0.625 (0.047)	10.398 (6.479)	25.660 (10.662)
5	2	**0.619** (0.062)	0.716 (0.139)	9.820 (4.933)	22.112 (6.591)
5	3	**0.591** (0.053)	0.725 (0.101)	10.867 (7.910)	22.645 (15.294)
5	4	**0.624** (0.063)	0.914 (0.126)	9.895 (3.905)	22.783 (7.374)
5	5	**0.593** (0.049)	0.970 (0.152)	10.747 (6.898)	26.816 (11.376)
8	1	**0.619** (0.053)	0.635 (0.059)	11.701 (8.661)	20.801 (6.050)
8	2	**0.624** (0.053)	0.709 (0.085)	9.359 (4.336)	23.228 (11.423)
8	3	**0.617** (0.059)	0.820 (0.084)	11.228 (6.391)	25.676 (11.703)
8	4	**0.623** (0.073)	0.976 (0.112)	9.612 (3.590)	29.258 (15.906)
8	5	**0.637** (0.068)	1.183 (0.202)	12.070 (10.618)	27.503 (10.582)
10	1	**0.616** (0.055)	0.644 (0.060)	8.799 (4.609)	20.105 (7.130)
10	2	**0.620** (0.050)	0.724 (0.075)	10.566 (6.501)	22.987 (10.152)
10	3	**0.611** (0.071)	0.850 (0.105)	11.330 (7.995)	24.671 (9.714)
10	4	**0.622** (0.054)	1.080 (0.117)	9.451 (2.696)	25.199 (9.725)
10	5	**0.619** (0.047)	1.343 (0.224)	13.740 (12.322)	23.771 (8.776)
15	1	**0.657** (0.073)	0.690 (0.067)	9.745 (4.586)	21.886 (8.389)
15	2	**0.705** (0.099)	0.797 (0.098)	12.448 (10.179)	23.652 (10.709)
15	3	**0.652** (0.089)	0.967 (0.095)	9.503 (4.188)	23.112 (6.335)
15	4	**0.648** (0.074)	1.194 (0.143)	8.747 (4.396)	28.678 (16.902)
15	5	**0.634** (0.067)	1.468 (0.219)	10.498 (3.959)	23.521 (7.166)

whether the data contains outliers or not. Furthermore, the performance comparison in terms of R^2 and Median Absolute Error (MAE) are provided in Tables 1 and 2 in Section 9 of the Supplementary Materials.

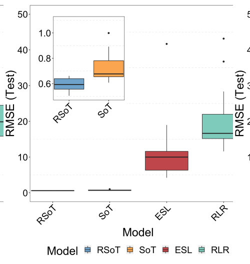
Moreover, Figure 2 shows the effect of the percentage of outliers, q, when $\nu = 3$. The figures show that as q increases, the performance of RSoT is the least affected, preserving similar performance over different percentages. However, the performance of all the benchmarks worsens substantially as q becomes larger. Figure 3 shows the effect of the mean of outlier distribution, ν, when $q = 8\%$. The figures demonstrate that the performance of RSoT is the least impacted by different levels of outlier means, whereas the performance of the benchmarks worsens dramatically as ν increases.

6.2 Simulation II: Poisson Distribution

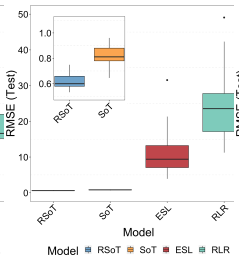
In this simulation study, we evaluate the performance of the proposed method when the target given the input data follows Poisson distribution, i.e., $y_i|\mathcal{X}_i \sim \text{Poisson}(\lambda_i)$, where λ_i is the rate parameter ($i = 1, \ldots, N$). We construct $\mathcal{X}_i$ and $\mathbf{U}_i$ similar to the first simulation.



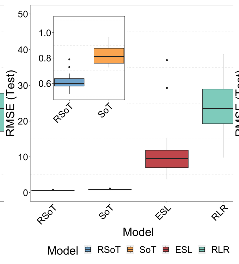
(a) $q = 3\%$



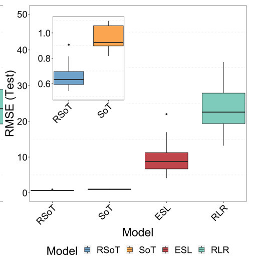
(b) $q = 5\%$



(c) $q = 8\%$

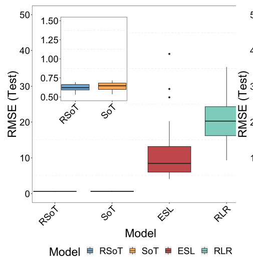


(d) $q = 10\%$

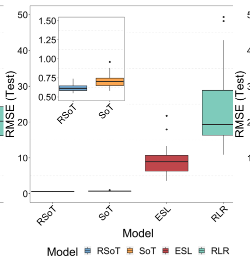


(e) $q = 15\%$

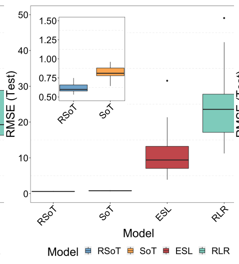
Figure 2: The effect of the percentage of outliers, q, when $\nu = 3$ in Simulation I



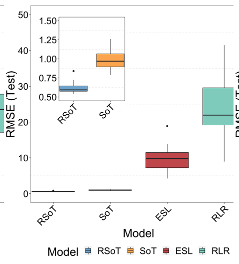
(a) $\nu = 1$



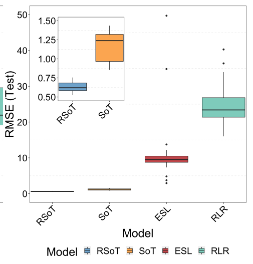
(b) $\nu = 2$



(c) $\nu = 3$



(d) $\nu = 4$



(e) $\nu = 5$

Figure 3: The effect of the mean of outlier distribution, ν, when $q = 8\%$ in Simulation I

Particularly, we generate $\mathbf{x}_i \in \mathbb{R}^{I_1 I_2 I_3}$ from multivariate normal distribution with mean $\boldsymbol{\mu}_x$ and the covariance matrix $\boldsymbol{\Sigma}$, i.e., $\boldsymbol{x}_i \sim \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma})$. The $(i,j)^{th}$ entry of $\boldsymbol{\Sigma}$ is $\rho^{|i-j|}$. $\mathcal{X}_i$ is then constructed by reshaping $\boldsymbol{x}_i$. $\boldsymbol{U}_i \in \mathbb{R}^{I_i \times R}, (i = 1, 2, 3)$, are simulated elementwise from a normal distribution with mean zero and variance σ_u^2, i.e., $\boldsymbol{U}_i \sim \mathcal{N}(0, \sigma_u^2)$. $\mathcal{B}$ is then constructed as $\mathcal{B} = \sum_{r=1}^{R} \boldsymbol{u}_1^{(r)} \circ \boldsymbol{u}_2^{(r)} \circ \boldsymbol{u}_3^{(r)}$. Then, we generate the rate parameter using log link as follows: $\lambda_i = \exp\{k\langle \mathcal{B}, \mathcal{X}_i \rangle + \mu\}$, where μ is the intercept and k is the multiplier to scale the inner product. We randomly select $(100 - q)$ % of the instances and simulate the responses as: $y_i \sim \text{Poisson}(\lambda_i)$, where q is the percentage of outliers. For the remaining q % of the instances, we simulate the responses as: $y_i \sim \text{Poisson}(\lambda_i + c\lambda_i)$, where c is the level of change in the rate parameter for the outliers. These instances are considered as outliers.

When constructing data, we set $\boldsymbol{\mu}_x = \mathbf{0.1} \in \mathbb{R}^{I_1 I_2 I_3}$, $\sigma_u = 0.5$, $\rho = 0.001$, $I_1 = 10, I_2 = 10, I_3 = 10$, $R = 2$ and $\mu = 1.5$. We simulate training data sets of size $N = 315$ with $c \in \{2, 3, 4, 5\}$ and $q \in \{0\%, 3\%, 5\%, 8\%, 10\%, 15\%\}$. We train models using the simulated data with the proposed method and benchmarks (i.e., SoT, ESL-LASSO, PCA-RLR, RLR). For the RSoT method, we select the rank from the set $\{2, 3, 4\}$, and α from the set $\{2, 1.414, 1, 0.707, 0.5, 0.353\}$ (i.e., $\frac{4}{2^{k/2}}$, $k = 2, \ldots, 7$) by using cross-validation as explained in Section 4.3, similar to Simulation I. To evaluate the performance of each method, we generate a test data set with 135 samples without any outliers. Our performance measure is Weighted Root Mean Squared Error (wRMSE) which is computed as wRMSE $= \sqrt{\frac{\sum_{i=1}^{N}(y_i - \widehat{y}_i)^2}{N\widehat{y}_i}}$, where y is the actual target, and $\widehat{y}$ is the predicted value. In Poisson distribution, the variance of the random variable (r.v.) is equal to the mean of the r.v., which is the rate, λ. Therefore, as the mean of the r.v. increases, its variance also increases. The proposed performance measure (wRMSE) adjusts for the increase in the variance as the mean estimate for the target increases. That is, it adjusts the squared deviations, $(y_i - \widehat{y}_i)^2$, with the mean estimate, $\widehat{y}_i$, $(i = 1, \ldots, N)$. In each scenario, we compare the performance of RSoT with the benchmarks in terms of wRMSE for different values of λ (identified by c) and the percentage of outliers (denoted by q).

Table 2 reports the mean and the standard deviation (shown in parenthesis) of wRMSE for each scenario. In addition, the average and the standard deviation of minimum error that can be achieved are reported in the last column. The minimum error is calculated by $\sqrt{\frac{\sum_{i=1}^{N}(y_i - \lambda_i)^2}{N\lambda_i}}$, where λ_i is the true rate parameters for i^{th} sample. Note that ESL-LASSO and RLR methods have Gaussian assumptions for the parameter estimation. Thus, these models may violate the non-positive mean estimation assumption, i.e., their estimated value could be non-positive ($\widehat{y}_i \leq 0$). To handle this situation in the performance evaluation, we cap the non-positive predicted values with one from below. Therefore, the denominator of the wRMSE for those samples is one, meaning that we do not incur an additional multiplying effect on the error term for those samples.

Table 2 shows that RSoT demonstrates superior performance compared to ESL-LASSO and RLR under all the outlier magnitudes (c) and the percentage of outliers (q). Moreover, RSoT outperforms SoT in almost all the scenarios, and it gives similar performance when there is no outlier and when the percentage and the level of outliers are the lowest. For instance, when $q = 10\%$ and $c = 4$, the wRMSE obtained by the proposed method is 1.628, whereas the wRMSEs are 2.191, 5.539, and 6.071 for SoT, ESL-LASSO, and RLR, respectively. These results indicate the robustness of RSoT when there exist outliers and show that the proposed method still has the predictive power of SoT when there is no outlier. The boxplots showing the effect of the percentage of outliers, q, and the level of

Table 2: Simulation II (Poisson Case): Comparison between the proposed method (RSoT) and the benchmarks in terms of wRMSE.

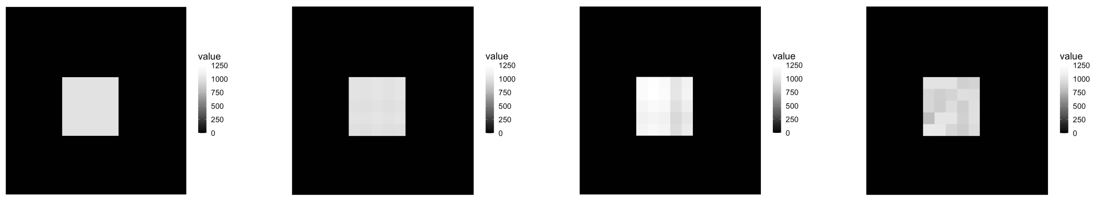
q (%)	c	RSoT	SoT	ESL-LASSO	RLR	Minimum Error
0	0	1.428 (0.189)	**1.375** (0.209)	6.996 (5.345)	4.917 (1.239)	1.001 (0.048)
3	2	1.488 (0.262)	**1.420** (0.175)	5.413 (2.175)	5.173 (1.770)	1.004 (0.057)
3	3	**1.490** (0.228)	1.596 (0.220)	7.667 (6.781)	5.566 (1.592)	1.010 (0.048)
3	4	**1.424** (0.138)	1.599 (0.184)	5.194 (1.733)	5.328 (1.615)	0.996 (0.055)
3	5	**1.487** (0.251)	1.984 (0.444)	7.054 (6.034)	5.537 (1.030)	1.015 (0.059)
5	2	**1.469** (0.193)	1.512 (0.233)	5.886 (1.790)	5.094 (0.851)	0.997 (0.056)
5	3	**1.501** (0.264)	1.637 (0.192)	5.924 (1.586)	5.670 (1.209)	0.983 (0.070)
5	4	**1.546** (0.289)	1.861 (0.364)	5.651 (1.868)	5.522 (0.719)	0.984 (0.056)
5	5	**1.494** (0.286)	1.980 (0.247)	5.761 (2.300)	6.179 (1.296)	0.997 (0.057)
8	2	**1.569** (0.208)	1.670 (0.197)	5.553 (2.176)	5.168 (0.714)	1.004 (0.069)
8	3	**1.606** (0.336)	1.929 (0.292)	5.058 (1.066)	5.493 (0.882)	1.000 (0.056)
8	4	**1.641** (0.293)	2.112 (0.397)	7.287 (4.231)	5.777 (0.876)	0.998 (0.077)
8	5	**1.564** (0.332)	2.176 (0.382)	10.196 (10.690)	5.860 (0.723)	0.994 (0.058)
10	2	**1.635** (0.302)	1.758 (0.287)	7.194 (3.822)	5.834 (1.264)	0.980 (0.061)
10	3	**1.701** (0.273)	2.002 (0.404)	8.413 (4.519)	5.888 (0.946)	0.991 (0.070)
10	4	**1.628** (0.390)	2.191 (0.438)	5.539 (2.409)	6.071 (0.760)	0.971 (0.050)
10	5	**1.695** (0.377)	2.457 (0.433)	7.415 (6.387)	6.374 (0.889)	0.989 (0.069)
15	2	**1.735** (0.276)	1.752 (0.222)	6.437 (3.776)	5.718 (1.578)	0.986 (0.056)
15	3	**1.918** (0.366)	2.071 (0.353)	8.202 (7.355)	6.553 (1.304)	0.976 (0.053)
15	4	**1.953** (0.372)	2.389 (0.356)	7.562 (6.612)	6.180 (1.068)	1.007 (0.066)
15	5	**1.762** (0.452)	2.577 (0.392)	9.164 (8.338)	7.178 (1.528)	0.980 (0.057)

change in the rate parameter of outliers, c, can be found in Section 5 of Supplementary Materials. Furthermore, the performance comparison in terms of Median Absolute Error (MAE) is provided in Table 3 in Section 10 of the Supplementary Materials.

6.3 Simulation III: Gaussian Distribution with Sparsity

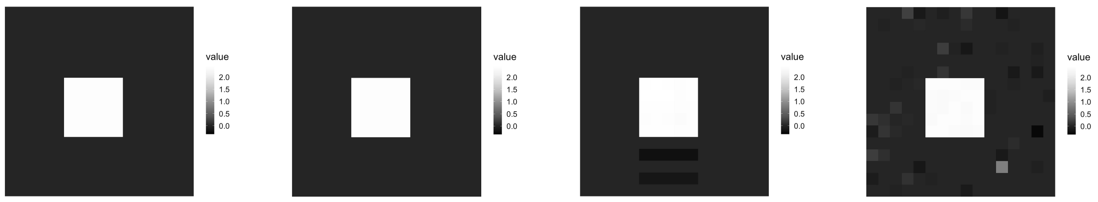
In this simulation study, we evaluate the performance of the proposed method when the tensor of parameters is sparse. For this purpose, we follow the simulation study in (Zhou et al., 2013). First, we employ the procedure described in Simulation I to construct input tensors. Particularly, we simulate $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2}$ with $I_1 = I_2 = 16$. Next, we construct the tensor of parameters $\mathcal{B} \in \mathbb{R}^{I_1 \times I_2}$ by setting $\mathcal{B}_{ij} = 10$ if $6 \leq i, j \leq 10$ and $\mathcal{B}_{ij} = 0$, otherwise. Hence, the tensor of parameters is sparse on its peripheral regions. Next, we simulate the responses by $y_i = \langle \mathcal{B}, \mathcal{X}_i \rangle + e_1$, where e_1 is a random error simulated from a Gaussian distribution with mean zero and variance σ_1^2. Then, we randomly select q % of the instances and add outliers such that $y_i := y_i + \beta y_i$, where β is the proportion of the target value added to the outlier instances.

We simulate training data sets of size $N = 315$ with $\sigma_1 = 1$, $\beta \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$, and $q \in \{0\%, 3\%, 5\%, 8\%, 10\%, 15\%\}$. We train a model using each method (i.e., RSoT, SoT, RLR). That is, we include sparsity penalty to the RSoT, SoT, and RLR approaches to perform variable selection. For RSoT and SoT, we select the sparsity penalty parameter, as explained in Section 4.3. We do 10-fold cross-validation to select the sparsity penalty parameter for RLR.For RSoT, we select α from the set $\{2, 1.414, 1, 0.707, 0.5, 0.353\}$ (i.e., $\frac{4}{2^{k/2}}$, $k = 2, \ldots, 7$) by using cross-validation as explained in Section 4.3 similar to Simulations I and II. To evaluate the performance of the RSoT and the benchmarks in terms of RMSPE, we generate a test data set with 135 samples without any outliers. We repeat this procedure 25 times to obtain the mean and the standard deviation of RMSPE. Figures 4 and 5 show examples of the true and the estimated parameters by each method when $q = 10\%$ and $\beta = 0.5$. These figures show the parameters using two different scaling for more clear visualization. As illustrated, the RLR estimation is inferior compared to RSoT and SoT. The estimated parameters deviate from the true values inside (Figure 4) and outside (Figure 5) of the light square. Furthermore, the SoT overestimates the value of the parameters inside the light square (see Figure 5) due to the lack of robustness. Table 3 reports the mean (standard deviation) of RMSPE for the proposed and benchmark methods for different percentages of outliers (q) and outlier levels (β). As it is reported, the proposed method demonstrates superior performance in the prediction of outputs, indicating the robustness of the proposed method in estimating sparse parameters. For instance, when $q = 8\%$ and $\beta = 0.3$, the RMSPE obtained by the proposed method is 1.034, whereas the SoT and RLR result in higher errors of 2.179 and 3.079, respectively. Moreover, when there is no outlier, the RSoT achieves the performance of the state-of-art tensor regression model. We also tested ESL-LASSO in the sparse setting, but it performed very poorly compared to the RSoT and the other benchmarks. Thus, we omit the results for ESL-LASSO in the table. The boxplots showing the effect of the percentage of outliers, q and the level of change in the rate parameter of outliers, c, can be found in Section 5 of Supplementary Materials. Furthermore, the performance comparison in terms of R^2 and Median Absolute Error (MAE) are provided in Tables 4 and 5 in Supplementary Results (Section 11).



(a) True (b) RSoT (c) SoT (d) RLR

Figure 4: Simulation III (Gaussian Case with Sparsity): True and estimated tensor of parameters (with cubic-scaling for illustration) by each method when $q = 10\%$ and $\beta = 0.5$.



(a) True (b) RSoT (c) SoT (d) RLR

Figure 5: Simulation III (Gaussian Case with Sparsity): True and estimated tensor of parameters (with log-scaling for illustration) by each method when $q = 10\%$ and $\beta = 0.5$.

7 Case Study

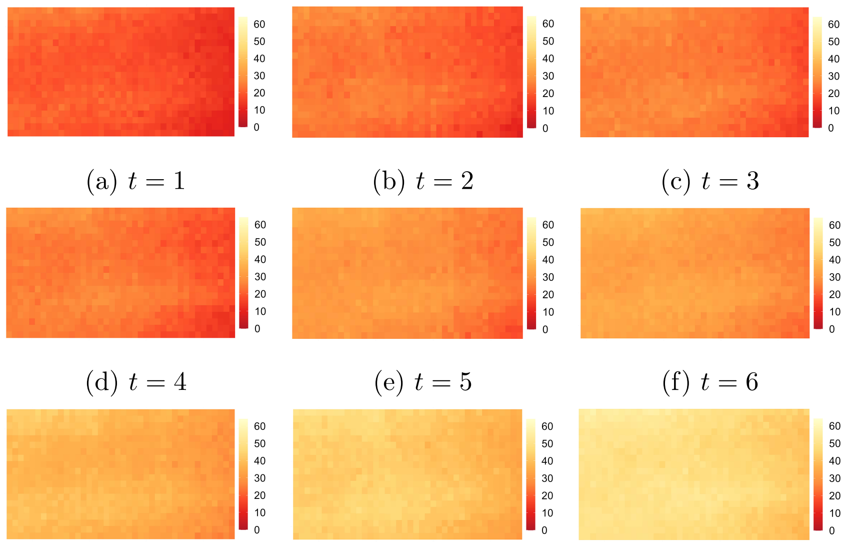
In this case study, we evaluate the performance of the proposed method in predicting the failure time of a rotating machinery given infrared thermal degradation images. Accelerated degradation tests were performed on rolling element thrust bearings using the experimental test bed described in Gebraeel et al. (2009). Test bearings were run from brand new until failure. Vibration signals were collected to monitor the health of the system and used as a proxy to the failure time of bearings. Once the amplitude of vibration frequencies exceed a pre-specified threshold based on ISO standards for machine vibration, a failure is recorded. Meanwhile, an FLIR T300 infrared camera captured thermal images of the bearing over the duration of the test. The images are 40×20 and are stored every 10 seconds illustrating the temperature signature of the degraded part over time. Four different experiments were run to failure. Each experiment resulted in an image stream containing 375, 611, 827, and 1478 images, respectively. Re-sampling is performed over the original four image streams,

Table 3: Simulation III (Gaussian Case with Sparsity): Comparison between the proposed method (RSoT) and the benchmarks in terms of RMSPE when the tensor of parameters is sparse.

q (%)	β	RSoT	SoT	RLR
0	0	**1.034** (0.061)	1.069 (0.081)	1.283 (0.087)
3	0.1	**1.030** (0.070)	1.106 (0.084)	1.236 (0.068)
3	0.2	**1.017** (0.061)	1.277 (0.155)	1.835 (0.264)
3	0.3	**1.011** (0.057)	1.484 (0.202)	2.299 (0.264)
3	0.4	**1.006** (0.060)	1.822 (0.436)	2.795 (0.471)
3	0.5	**1.026** (0.057)	2.014 (0.531)	3.334 (0.741)
5	0.1	**1.026** (0.058)	1.172 (0.059)	1.511 (0.125)
5	0.2	**1.019** (0.053)	1.383 (0.139)	2.023 (0.229)
5	0.3	**0.998** (0.054)	1.854 (0.431)	2.649 (0.355)
5	0.4	**1.026** (0.057)	2.215 (0.329)	3.259 (0.408)
5	0.5	**1.034** (0.059)	2.513 (0.488)	3.717 (0.583)
8	0.1	**1.051** (0.063)	1.245 (0.099)	1.601 (0.161)
8	0.2	**1.015** (0.055)	1.695 (0.224)	2.277 (0.220)
8	0.3	**1.034** (0.057)	2.179 (0.344)	3.079 (0.394)
8	0.4	**1.017** (0.052)	2.705 (0.483)	3.560 (0.419)
8	0.5	**1.024** (0.064)	3.330 (0.612)	4.313 (0.566)
10	0.1	**1.049** (0.072)	1.319 (0.122)	1.728 (0.202)
10	0.2	**1.047** (0.070)	1.857 (0.248)	2.539 (0.346)
10	0.3	**1.020** (0.060)	2.474 (0.430)	3.215 (0.506)
10	0.4	**1.012** (0.070)	3.079 (0.523)	3.888 (0.481)
10	0.5	**1.029** (0.062)	3.765 (0.659)	4.478 (0.474)
15	0.1	**1.065** (0.054)	1.433 (0.118)	1.812 (0.182)
15	0.2	**1.058** (0.067)	2.282 (0.308)	2.651 (0.341)
15	0.3	**1.053** (0.069)	3.041 (0.474)	3.455 (0.355)
15	0.4	**1.024** (0.063)	4.261 (0.816)	4.140 (0.554)
15	0.5	**1.055** (0.072)	5.113 (1.088)	4.834 (0.606)

to produce a total of 284 image streams. For more details regarding this experiment please see Fang et al. (2019). Figure 6 illustrates a sequence of images collected over time.

One challenge is the variable length (duration) of the collected degradation signals, which causes different sizes of input tensors. In this case study, we take the length of the signal to be τ and only consider the first τ images of the samples that failed after time τ. The underlying reason is that, in real applications, when predicting the failure time of a machine, we can only exploit the information in the image streams collected from the beginning until the current time point t. For example, at $t = 10$, we only have access to the image streams until $t = 10$ ($\tau = 10$). As time passes, we will have access to more images, which will potentially help improve the prediction performance since more information becomes available. Hence, each image stream is truncated by only keeping images observed in the time interval $[0,\tau]$. Therefore, the input tensors are $40 \times 20 \times \tau$. The lifetime T_i is associated to each of the input tensors $\mathcal{X}_i$. Note that the sample size of the original data is very small and contains only four



(a) $t = 1$ (b) $t = 2$ (c) $t = 3$

(d) $t = 4$ (e) $t = 5$ (f) $t = 6$

(g) $t = 7$ (h) $t = 8$ (i) $t = 9$

Figure 6: Illustration of a sequence of thermal images captured by an infrared camera.

actual tests due to the high cost of running degradation experiments. Therefore, we followed the resampling strategy of (Fang et al., 2019) to produce a dataset with a larger sample size (dataset A). Figure 7a illustrates the relationship between the first principal component score of the images and the failure time when $\tau = 10$. Data does not show the existence of outliers with respect to this specific extracted feature. However, this does not necessarily mean that the original data does not contain outliers. In real applications, we usually do not know if outliers exist in the data. Therefore, it is crucial to have a model which preserves its performance when trained with data that may or may not contain outliers. We used dataset A to evaluate the performance of the method when no information about the existence of outliers was available. Furthermore, to evaluate the performance of our method under the scenario that outliers certainly exist, we performed a slightly different resampling strategy from (Fang et al., 2019) to produce outliers by changing the frequency of the resampling in 5 % of image streams from the original streams (dataset B). That is, we distort the temporal pattern of an image stream to create outliers. More details about this resampling strategy are provided in the Supplementary Materials. Figure 7b demonstrates the relationship between the first principal component score of the images and the failure time with additional outliers shown in red circles.

We evaluate the performance of the proposed method, both with and without additional

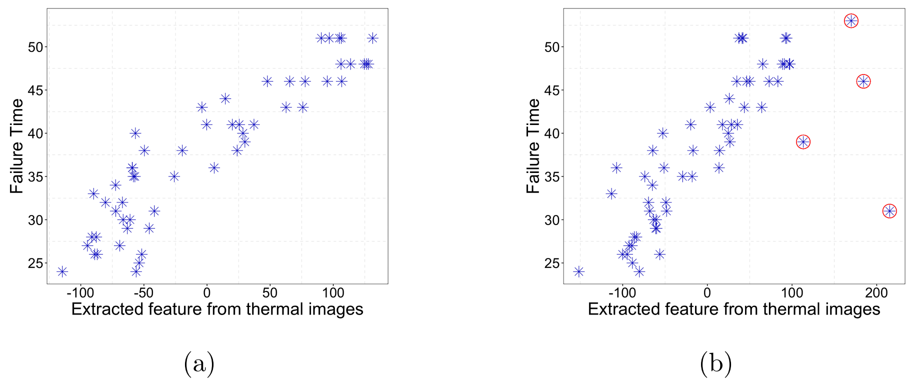


Figure 7: Illustration of extracted feature from (a) thermal image sequences with no outliers and (b) from thermal image sequences with outliers.

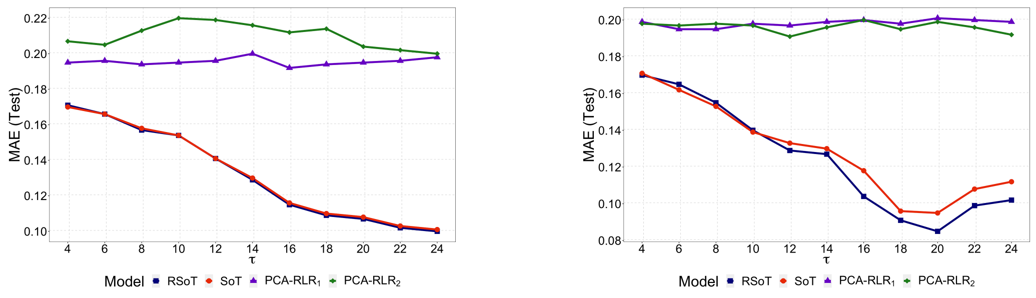
outliers. We aim to show the effect of the existence of outliers on the performance of the proposed method. Since the data contains outliers, we evaluate the performance of the proposed method in terms of median absolute error (MAE) calculated as MAE $=$ $\text{median}_i(|\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i|)$, where $\boldsymbol{y}_i$ is the actual value and $\hat{\boldsymbol{y}}_i$ is the predicted value. In the case study, the test set contains outliers, unlike in the simulation studies. Therefore, choosing a performance measure that is robust to outliers is important.

To evaluate the performance of RSoT and SoT, we first perform Multilinear Principal Component Analysis (MPCA) on the image streams for enhanced computation by following (Fang et al., 2019). The MPCA exploits the high spatial correlation structure in the thermal images to reduce the input tensor dimensions. Proposition 1 in (Fang et al., 2019) shows that original high-dimensional tensors and their low-dimensional projections result in similar output predictions. After this step, we obtain image streams of size $2 \times 2 \times \tau$. The dimensions are chosen such that around 95 % of the variance in the image stream tensors is explained. We compare the proposed method to the SoT and RLR. As RLR fails to provide estimations due to the high dimensionality of the data ($n < p$), we first perform Principal Component Analysis (PCA) on the vectorized image streams. First, we obtain the principal component scores such that 95 % variance is explained. Then, we evaluate the performance of RLR on this data of extracted scores, denoted as PCA-RLR$_1$. In the second case, we select the number of the principal component scores as $2 \times 2 \times \tau = 4\tau$. Thus, in this case, the input data for RLR has the same size as the input data for RSoT and SoT. Then, we evaluate the performance of RLR on this input data, denoted as PCA-

RLR$_2$. We divide 70 % of the data for training and 30 % for testing. We perform the comparison at different values of τ, $\tau = \{4, 6, 8, \ldots, 24\}$. The larger the τ is, the more information is available for predicting the failure time. For the RSoT method, we select α from the set $\{0.125, 0.0884, 0.0625, 0.0442, 0.0312, 0.0221, 0.0156, 0.0110, 0.0078, 0.0055\}$ (i.e., $\frac{4}{2^{k/2}}$, $k = 10, \ldots, 19$) by using cross-validation as explained in Section 4.3. Our cross-validation performance measure is truncated RMSE, calculated based on 90 % of the instances of the least squared errors. We also test ESL-LASSO on this dataset. Although we carefully tune the hyperparameters which control the robustness and regularization, it performs very poorly compared to RSoT and the other benchmarks. Therefore, we do not report the results for ESL-LASSO.

One challenge in this case study is that failure times do not follow the Gaussian distribution. Fang et al. (2019) show that log-normal distribution is one of the distributions that perform well for this data. Since they are generalized models, RSoT and SoT can handle log-normal distribution. However, RLR can only handle Gaussian distribution. To make the comparison fair for all the benchmarks, we transform the failure times using log transformation and then test PCA-RLR$_1$ and PCA-RLR$_2$. Note that we report the performance measures on a log scale. Figure 8a illustrates the performance of each method in terms of MAE when there are no additional outlier streams. As it is depicted, the RSoT outperforms the PCA-RLR$_1$ and PCA-RLR$_2$ for all τ values. Furthermore, RSoT performs very similarly to the SoT. Also, the larger the value of τ, the more accurate the estimations become. Moreover, Figure 8b shows the performance of each method when few outlier streams are added additionally. In this setting, RSoT outperforms all the benchmark. These results show that RSoT provides superior performance when there are outliers and does not lose the power of SoT when there are no outliers. The superior performance of RSoT is because it captures the spatio-temporal correlation structures in the image streams, and it is robust to the outlier instances. Therefore, it is advantageous to use RSoT as in most real-life cases, it is unknown whether outliers exist in a dataset.

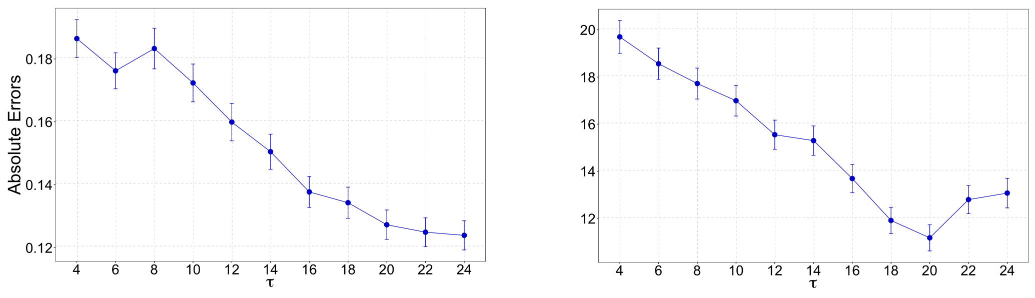
Figures 9a and 9b depict the prediction errors of RSoT with a 95 % confidence interval both when there is no outlier and when there are outliers, respectively. Both figures show that as image streams contain more images, i.e., as τ gets larger, the prediction errors decrease as expected because more information can be used to predict the failure time of



(a) (b)

Figure 8: Performance comparison of all the methods (a) without outlier and (b) with outliers (b). RSoT outperforms the benchmarks and has the same power as SoT when there are no outliers.

the machines. Furthermore, as τ gets larger, the prediction error intervals decrease since using more images diminishes the uncertainty.



(a) (b)

Figure 9: Prediction errors of RSoT with 95 % confidence interval when (a) there are no outliers and (b) when there are outliers

8 Discussion

In the paper, although we only show the theoretical results for the normal distribution in the paper, our proposed methodologies with Lq-likelihood function work for the generalized tensor regression model. In particular, we have conducted simulations and case studies to show our method also yields adequate performance for non-Gaussian distribution. We only conduct theoretical analysis for Gaussian distribution because, under the Gaussian assumption, our general robust estimator will reduce to the classical M-estimator with a special exponential square loss $\rho_\alpha(.)$. This will allow us to apply theoretical results of general robust M-estimators to obtain our asymptotic results. This is also why one can see similarities between our results and (Zhou et al., 2013) since both estimators (Lq under Gaussian and log-likelihood) belong to the family of M-estimators. Unfortunately, it is still very challenging to derive the theoretical asymptotic properties for the generalized tensor regression model with the Lq-likelihood function, as the results from the family of M-estimators do not apply when the underlying distribution is not Gaussian. As far as we know, the only asymptotic results for the maximum Lq-likelihood estimator were derived in (Ferrari and Yang, 2010). However, that paper only focuses on the robust point estimation problem when data are i.i.d. with the exponential family. For our generalized tensor regression model, under the low-rank assumption where $\mathcal{B}_0 = [[\boldsymbol{U}_{01}, \boldsymbol{U}_{02}, \cdots, \boldsymbol{U}_{0m}]]$, the parameters $\boldsymbol{U}_{0i}$ are not the parameters of any exponential family distributions. Moreover, when the input tensors $\mathcal{X}_i$ are fixed, the observed data $y_1, y_2, \cdots, y_n$ are not i.i.d, which means the elements $\frac{1-(f_{\theta_i}(y_i))^\alpha}{\alpha}$ in our objective function are not i.i.d. Therefore, extending such results in (Ferrari and Yang, 2010) to our generalized tensor regression model remains a very challenging task and new techniques may be developed to solve the problem. Studying these challenges is an important future direction for us to pursue.

The major contribution of our paper is to propose a new framework for generalized tensor regression using the general Lq-likelihood to achieve robustness. In many cases, it is not known whether the data contains outliers, and using robust algorithms allows for achieving models with strong prediction performance. To complete the framework, we introduced algorithms to estimate the robust estimator, derived some theoretical properties of our estimator, provided guidelines on the selection of tuning parameters, and performed multiple simulations and case studies to illustrate the superior performance of our proposed method.

We use CP decomposition in the proposed tensor regression model. CP decomposition has also commonly been used in tensor regression, and its effectiveness has been shown in (Zhou et al., 2013; Fang et al., 2019). The main advantage of CP decomposition over Tucker decomposition is that it has a fewer tuning parameters. More specifically, CP decomposition requires selecting a single rank. Since our proposed method contains other tuning parameters other than rank (e.g., α), the CP decomposition has been selected to reduce the model tuning efforts. However, we agree that Tucker decomposition would be an alternative decomposition technique which has been shown to work well in high-dimensional settings, which can also be a future direction for us.

Furthermore, Newton's search method is not an integral part of the proposed method. The search method used in parameter estimation can be determined by the user depending on the resources they have. Particularly, in the proposed method, the complexity to calculate the gradient is $O(N(R^2 I_d + R\Pi_{i=1}^m I_i))$, whereas the complexity to calculate the Hessian matrix is $O(N(R^2 I_d + R\Pi_{i=1}^m I_i + I_d^2))$. Thus, the computational complexity of parameter estimation with Newton's search method is $O(\sum_{r=1}^R \sum_{d=1}^m n_d^{(r)}(N(R^2 I_d + R\Pi_{i=1}^m I_i + I_d^2) + \frac{1}{3} I_d^3))$, where $n_d^{(r)}$ is the number of iterations to update $\mathbf{u}_d^{(r)}$ and when Cholesky decomposition is used for inverse calculation of the Hessian matrix. For larger-scale problems where the inverse calculation of the Hessian matrix is too costly, the proposed framework can easily be adjusted to use other optimization techniques, such as quasi-Newton, and first- and zero-order techniques.

9 Conclusion

This paper proposes a robust estimation approach to the generalized linear scalar-on-tensor regression model, which is widely used to capture high-dimensional structured data such as images and profiles. To achieve robustness, Lq-likelihood loss is used and the corresponding robust estimator is computed through an alternating least squares algorithm with a second-order search method. A large number of model parameters may result in overfitting, which is handled by introducing a low-rank constraint on the tensor of parameters. The efficacy of the proposed approach is evaluated using simulation and case studies. The simulation results under both Gaussian and Poisson distribution assumptions indicate that considering the robustness in the tensor model creates superior performance in comparison to the benchmarks that are either not robust or may not capture the structure of the data. The case study evaluates the performance of the method in predicting the failure time of a rotary machine given thermal images. Many future directions can be considered, including the existence of outliers and missing values within the input data.

10 Data Availability

To request access to the data used in the case study, one may contact the corresponding author of (Gebraeel et al., 2009), who owns the data.

Acknowledgements

This work has been partially supported by the National Science Foundation (NSF) award 2212878. Dr. Zhang's research is supported in part by NSF grant ECCS-2236565. We would like to thank Dr. Gebraeel for providing us with the case study data.

References

Barber, R. F., M. Reimherr, and T. Schill (2017). The function-on-scalar lasso with applications to longitudinal gwas. *Electronic Journal of Statistics 11*(1), 1351–1389.

Beaton, A. E. and J. W. Tukey (1974). The fitting of power series, meaning polynomials, illustrated on band-spectroscopic data. *Technometrics 16*(2), 147–185.

Chang, L., S. Roberts, and A. Welsh (2018). Robust lasso regression using tukey's biweight criterion. *Technometrics 60*(1), 36–47.

Chen, H., G. Raskutti, and M. Yuan (2019). Non-convex projected gradient descent for generalized low-rank tensor regression. *The Journal of Machine Learning Research 20*(1), 172–208.

Chen, Y., J. Goldsmith, and R. T. Ogden (2016). Variable selection in function-on-scalar regression. *Stat 5*(1), 88–101.

Dennis Jr, J. E. and R. E. Welsch (1978). Techniques for nonlinear least squares and robust regression. *Communications in Statistics-Simulation and Computation 7*(4), 345–359.

Fan, Z. and M. Reimherr (2017). High-dimensional adaptive function-on-scalar regression. *Econometrics and Statistics 1*, 167–183.

Fang, X., K. Paynabar, and N. Gebraeel (2019). Image-based prognostics using penalized tensor regression. *Technometrics 61*(3), 369–384.

Ferrari, D. and Y. Yang (2010). Maximum Lq-likelihood estimation. *The Annals of Statistics 38*(2), 753 – 783.

Gahrooei, M. R., H. Yan, K. Paynabar, and J. Shi (2021). Multiple tensor-on-tensor regression: An approach for modeling processes with heterogeneous sources of data. *Technometrics 63*(2), 147–159.

Gebraeel, N., A. Elwany, and J. Pan (2009). Residual life predictions in the absence of prior degradation knowledge. *IEEE Transactions on Reliability 58*(1), 106–117.

Guhaniyogi, R., S. Qamar, and D. B. Dunson (2017). Bayesian tensor regression. *The Journal of Machine Learning Research 18*(1), 2733–2763.

Huber, P. J. (1964). Robust estimation of a location parameter. *The Annals of Mathematical Statistics 35*(1), 73–101.

Kalogridis, I. and S. Van Aelst (2019). Robust functional regression based on principal components. *Journal of Multivariate Analysis 173*, 393–415.

Kanning, M., I. Kühling, D. Trautz, and T. Jarmer (2018). High-resolution UAV-based hyperspectral imagery for LAI and chlorophyll estimations from wheat for yield prediction. *Remote Sensing 10*(12), 2000.

Kowal, D. R. and D. C. Bourgeois (2020). Bayesian function-on-scalars regression for high-dimensional data. *Journal of Computational and Graphical Statistics 29*(3), 629–638.

Lambert-Lacroix, S. and L. Zwald (2011). Robust regression through the huber's criterion and adaptive lasso penalty. *Electronic Journal of Statistics 5*, 1015–1053.

Li, B., X. Xu, L. Zhang, J. Han, C. Bian, G. Li, J. Liu, and L. Jin (2020). Above-ground biomass estimation and yield prediction in potato by using UAV-based RGB and hyperspectral imaging. *ISPRS Journal of Photogrammetry and Remote Sensing 162*, 161–172.

Li, G., H. Peng, and L. Zhu (2011). Nonconcave penalized m-estimation with a diverging number of parameters. *Statistica Sinica 21*, 391–419.

Li, N., N. Gebraeel, Y. Lei, X. Fang, X. Cai, and T. Yan (2021). Remaining useful life prediction based on a multi-sensor data fusion model. *Reliability Engineering & System Safety 208*, 107249.

Li, X., D. Xu, H. Zhou, and L. Li (2018). Tucker tensor regression and neuroimaging analysis. *Statistics in Biosciences 10*(3), 520–545.

Maronna, R. A. and V. J. Yohai (2013). Robust functional linear regression based on splines. *Computational Statistics & Data Analysis 65*, 46–55.

Ogden, R. T., C. E. Miller, K. Takezawa, and S. Ninomiya (2002). Functional regression in crop lodging assessment with digital images. *Journal of Agricultural, Biological, and Environmental Statistics 7*(3), 389–402.

Wang, X., Y. Jiang, M. Huang, and H. Zhang (2013). Robust variable selection with exponential squared loss. *Journal of the American Statistical Association 108*(502), 632–643.

Wang, Y., H. Wang, D. Srinivasan, and Q. Hu (2019). Robust functional regression for wind speed forecasting based on sparse bayesian learning. *Renewable Energy 132*, 43–60.

Yan, H., K. Paynabar, and M. Pacella (2019). Structured point cloud data analysis via regularized tensor regression for process modeling and optimization. *Technometrics 61*(3), 385–395.

Zhang, A. R., Y. Luo, G. Raskutti, and M. Yuan (2020). Islet: Fast and optimal low-rank tensor regression via importance sketching. *SIAM Journal on Mathematics of Data Science 2*(2), 444–479.

Zhou, H., L. Li, and H. Zhu (2013). Tensor regression with applications in neuroimaging data analysis. *Journal of the American Statistical Association 108*(502), 540–552.