Tensor Gaussian Process with Contraction for Multi-Channel Imaging Analysis

Hu Sun [1] Ward Manchester [2] Meng Jin [3] Yang Liu [4] Yang Chen [1]

[1]Department of Statistics, University of Michigan, Ann Arbor [2]Department of Climate and Space Sciences and Engineering, University of Michigan, Ann Arbor [3]Solar & Astrophysics Lab, Lockheed Martin [4]W.W. Hansen Experimental Physics Laboratory, Stanford University

Background: Tensor Regression & Tensor Gaussian Process

In this project, we consider a regression problem where the scalar label $y \in \mathbb{R}$ is associated with m-mode tensor covariate $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots I_m}$. Classic parametric *scalar-on-tensor* regression model formulates the relationship between y and $\mathcal{X}$ as:

$$y = \langle \mathcal{W}, \mathcal{X} \rangle + \epsilon, \tag{1}$$

where $\mathcal{W} \in \mathbb{R}^{I_1 \times I_2 \times \ldots I_m}$ is the tensor regression coefficient and ϵ is the additive noise term and $\langle \cdot, \cdot \rangle$ is the tensor inner product. Typically, $\mathcal{W}$ is assumed to be low-rank and follow a rank-$(r_1, r_2, \ldots, r_m)$ Tucker decomposition:

$$\mathcal{W} = \mathcal{S} \times_1 \mathbf{U}_1^\top \times_2 \mathbf{U}_2^\top \times_3 \cdots \times_m \mathbf{U}_m^\top, \tag{2}$$

where $\mathcal{S} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_m}$ is the "core" tensor, with $\Pi_{k=1}^m r_k << \Pi_{k=1}^m I_k$, $\mathbf{U}_k \in \mathbb{R}^{r_k \times I_k}$ and $\times_k$ is the k^{th}-mode tensor mode product. See [1] for details on these tensor algebra concepts.

In [2], the regression model (1) with the assumption (2) is re-formulated as a tensor Gaussian Process (**Tensor-GP**) model:

$$y = f(\mathcal{X}) + \epsilon, \quad f(\cdot) \sim \text{GP}\left(0, k(\cdot, \cdot)\right), \tag{3}$$

where $k(\cdot, \cdot)$ is the *multi-linear* tensor kernel function:

$$k(\mathcal{X}_1, \mathcal{X}_2) = \mathbf{vec}\left(\mathcal{X}_1\right)^\top \left(\mathbf{U}_m^\top \mathbf{U}_m \otimes \mathbf{U}_{m-1}^\top \mathbf{U}_{m-1} \otimes \cdots \otimes \mathbf{U}_1^\top \mathbf{U}_1\right) \mathbf{vec}\left(\mathcal{X}_2\right), \tag{4}$$

where $\mathbf{vec}\left(\cdot\right)$ is the vectorization operator and $\otimes$ is matrix Kronecker product.

Methodology Overview

In this project, we consider $\mathcal{X}$ as an $H \times W \times C$ multi-channel imaging tensor with H, W, C as the height, width and number of channels, respectively. We expand the Tensor-GP in (3) into a two-step procedure (called **Tensor-GPST**):

1. (**Tensor Contraction**): we estimate a latent tensor $\mathcal{Z} \in \mathbb{R}^{h \times w \times C}$ with $h << H, w << W$ for $\mathcal{X}$;
2. (**Tensor GPR**): we then estimate the Tensor-GP regression model between y and $\mathcal{Z}$.

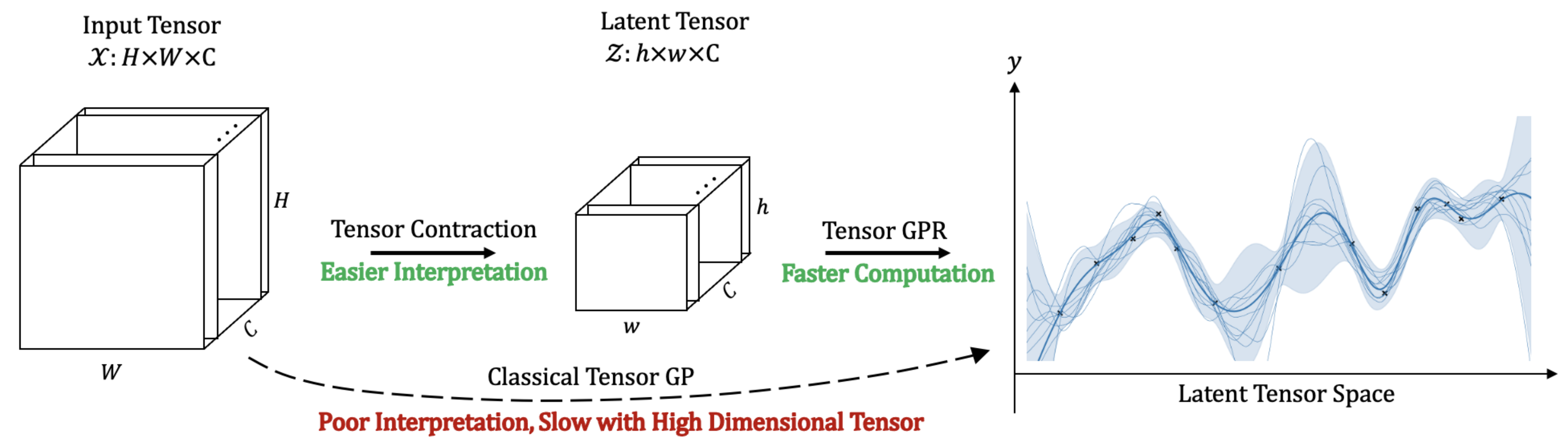


Figure 1. Tensor Contraction + Tensor GP Regression Procedure

Tensor Gaussian Process with Spatial Transformation (Tensor-GPST)

Given data $\{\mathcal{X}_i, y_i\}_{i=1}^N$, we propose the following framework:

$$y_i = f \circ g(\mathcal{X}_i) + \epsilon_i, \tag{Tensor-GPST}$$

$$\mathcal{Z}_i = g(\mathcal{X}_i) = \mathcal{X}_i \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{I}_C \tag{Tensor Contraction}$$

$$f(\mathcal{Z}_i) \sim \text{GP}\left(0, k(\cdot, \cdot)\right), k(\mathcal{Z}_i, \mathcal{Z}_j) = \mathbf{vec}\left(\mathcal{Z}_i\right)^\top \left(\mathbf{K}_3 \otimes \mathbf{K}_2 \otimes \mathbf{K}_1\right) \mathbf{vec}\left(\mathcal{Z}_j\right), \tag{Tensor GPR}$$

where we specify $\mathbf{K}_m = \mathbf{U}_m^\top \mathbf{U}_m, m = 1, 2, 3$. Equivalently, our model specifies a Tensor-GP:

$$y_i = h(\mathcal{X}_i) + \epsilon_i, \quad h(\cdot) \sim \text{GP}\left(0, \mathcal{K}(\cdot, \cdot)\right)$$

$$\mathcal{K}\left(\mathcal{X}_i, \mathcal{X}_j\right) = \mathbf{vec}\left(\mathcal{X}_i\right)^\top \left[\mathbf{K}_3 \otimes \left(\mathbf{B}^\top \mathbf{K}_2 \mathbf{B}\right) \otimes \left(\mathbf{A}^\top \mathbf{K}_1 \mathbf{A}\right)\right] \mathbf{vec}\left(\mathcal{X}_j\right) \tag{5}$$

Loss Function & Tensor Contraction with Total-Variation Penalty

We propose to minimize the following penalized negative marginal log-likelihood of $\mathbf{y}$ for parameter estimation:

$$L(\mathbf{y}|\mathbf{A}, \mathbf{B}, \mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3, \sigma) = \underbrace{\frac{1}{2}\log\left|\mathbf{K} + \sigma^2 \mathbf{I}_N\right| + \frac{1}{2}\mathbf{y}^\top \left(\mathbf{K} + \sigma^2 \mathbf{I}_N\right)^{-1} \mathbf{y}}_{\text{Negative Marginal Log-Likelihood } \ell(\mathbf{y}|\mathbf{A}, \mathbf{B}, \mathbf{K}_1, \mathbf{K}_2, \mathbf{K}_3, \sigma)} + \lambda \mathrm{R}(\mathbf{A}, \mathbf{B}), \tag{6}$$

where $\mathbf{K}_{N \times N}$ is the kernel gram matrix based on the kernel in (5) and $\mathrm{R}(\mathbf{A}, \mathbf{B})$ is a total-variation penalty over $\mathbf{A}, \mathbf{B}$. To see the exact form of $\mathrm{R}(\mathbf{A}, \mathbf{B})$, first consider the tensor contraction step:

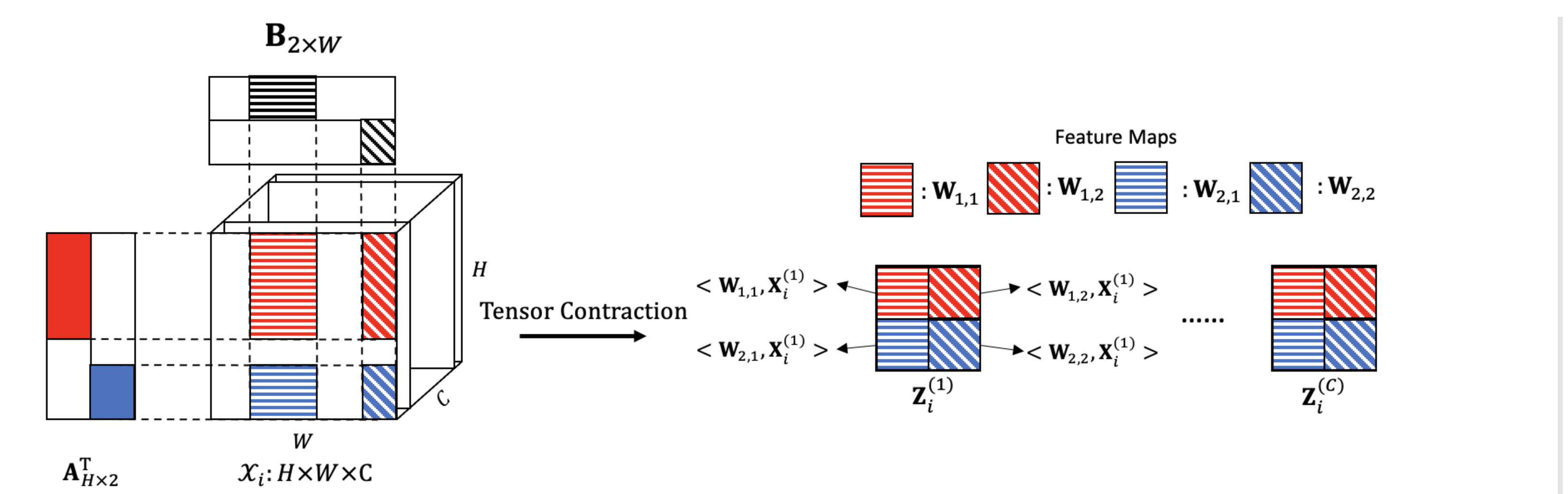


Figure 2. Breakdown of the Tensor Contraction Operation on the Input Imaging Data. Superscript (c) denotes the c^{th} channel of tensor.

Essentially, the $(s, t, c)^{\text{th}}$ entry of $\mathcal{Z}_i$ is computed via: $\mathcal{Z}_i(s, t, c) = \left\langle \mathbf{A}(s, :)^\top \mathbf{B}(t, :), \mathcal{X}_i(:, :, c) \right\rangle$, and let $\mathbf{W}_{st} = \mathbf{A}(s, :)^\top \mathbf{B}(t, :)$ be the "feature map" for the $(s, t)^{\text{th}}$ feature of the c^{th} channel of $\mathcal{X}_i$, we penalize its anisotropic total variation (TV) norm:

$$\|\mathbf{W}_{s,t}\|_{\text{TV}} = \sum_{i=1}^{H-1}\sum_{j=1}^{W} \left|\mathbf{W}_{s,t}(i+1, j) - \mathbf{W}_{s,t}(i, j)\right| + \sum_{i=1}^{H}\sum_{j=1}^{W-1} \left|\mathbf{W}_{s,t}(i, j+1) - \mathbf{W}_{s,t}(i, j)\right|, \tag{7}$$

which induces the penalty over $\mathbf{A}, \mathbf{B}$ as:

$$\sum_{s=1}^{h}\sum_{t=1}^{w} \|\mathbf{W}_{s,t}\|_{\text{TV}} = \|\nabla_x \mathbf{B}\|_1 \times \|\mathbf{A}\|_1 + \|\mathbf{B}\|_1 \times \|\nabla_x \mathbf{A}\|_1 := \mathrm{R}(\mathbf{A}, \mathbf{B}). \tag{8}$$

Estimating Algorithm: Alternating Proximal Gradient Descent

To minimize the loss in (6), we attempt to update the model parameters one at a time in the order of: $\mathbf{A} \to \mathbf{B} \to (\mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3) \to \sigma \to \mathbf{A} \to \ldots$. The gradients of $\ell(\cdot)$ can be easily computed since $\mathbf{K} = \widetilde{\mathbf{U}}\widetilde{\mathbf{U}}^\top$ where $\widetilde{\mathbf{U}} = \widetilde{\mathcal{X}}^\top \left(\mathbf{I}_C \otimes \mathbf{B} \otimes \mathbf{A}\right)^\top \left(\mathbf{U}_3 \otimes \mathbf{U}_2 \otimes \mathbf{U}_1\right)^\top$, with $\widetilde{\mathcal{X}} = \left[\mathbf{vec}\left(\mathcal{X}_1\right); \mathbf{vec}\left(\mathcal{X}_2\right); \ldots; \mathbf{vec}\left(\mathcal{X}_N\right)\right]$, and Woodbury identity can be used to compute $(\mathbf{K} + \sigma^2 \mathbf{I}_N)^{-1}$ without incurring a computational cost at $\mathcal{O}(N^3)$.

To update $\mathbf{A}$ at iteration i, we need to further consider $\mathrm{R}(\mathbf{A}, \mathbf{B})$ by applying the proximal operator to the gradient descent update $\widehat{\mathbf{A}}^{(i+\frac{1}{2})} = \widehat{\mathbf{A}}^{(i)} - \eta_i \partial \ell / \partial \mathbf{A}$:

$$\widehat{\mathbf{A}}^{(i+1)} = \mathbf{prox}_{\text{TV}}\left(\widehat{\mathbf{A}}^{(i+\frac{1}{2})}\right) = \arg\min_{\mathbf{A}} \left\{\frac{1}{2\eta_i}\left\|\mathbf{A} - \widehat{\mathbf{A}}^{(i+\frac{1}{2})}\right\|_{\text{F}}^2 + \lambda \mathrm{R}(\mathbf{A}, \widehat{\mathbf{B}}^{(i)})\right\}. \tag{9}$$

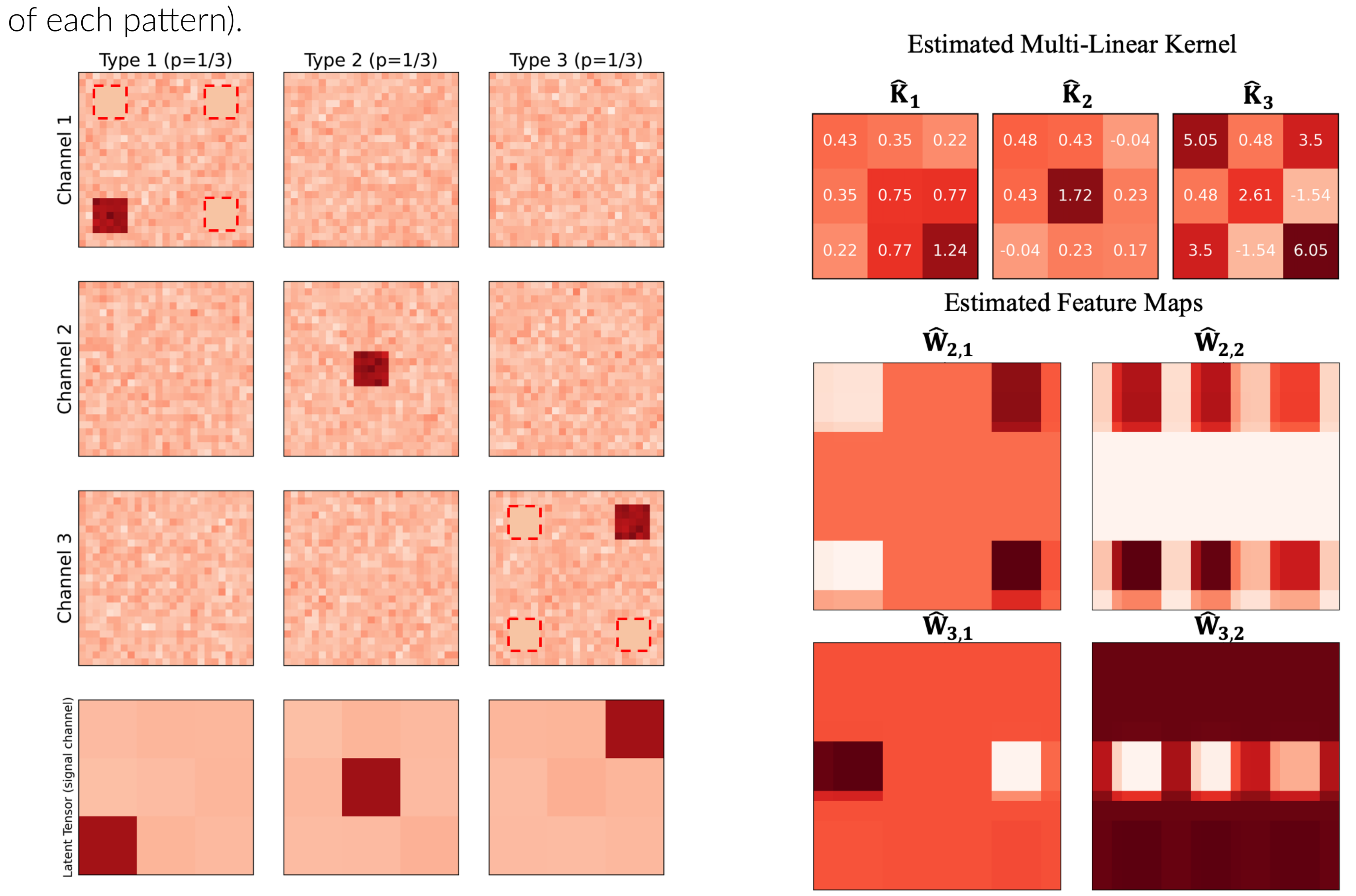
The proximal operator in (9) is essentially solving 1-D fused lasso problem for each row of $\mathbf{A}$:

$$\mathbf{A}^{(i+1)}(s, :) \leftarrow \arg\min_{\boldsymbol{\alpha} \in \mathbb{R}^H} \frac{1}{2\eta_i}\left\|\boldsymbol{\alpha} - \widehat{\mathbf{A}}^{(i+\frac{1}{2})}(s, :)\right\|_{\text{F}}^2 + \lambda_1 \cdot \sum_{j=2}^{H} |\boldsymbol{\alpha}(j+1) - \boldsymbol{\alpha}(j)| + \lambda_2 \cdot \|\boldsymbol{\alpha}\|_1, \quad s = 1, 2, \ldots, h,$$

where $\lambda_1 = \lambda\|\mathbf{B}^{(i)}\|_1, \lambda_2 = \lambda\|\nabla_x \mathbf{B}^{(i)}\|_1$. In our paper, we further prove that the algorithm converges to a local minimum with rate $\mathcal{O}(1/K)$, with K being the total number of iterations.

Simulation Experiment

We simulate a 3-channel imaging tensor dataset $\mathcal{X}_i$ of size $25 \times 25 \times 3$, and put a 5×5 signal block in one of the three channels, leading to three patterns of tensor data (see Figure 3a for samples of each pattern).



(a) Examples of Simulated Tensor Data. Dashed boxes are possible locations of signal blocks.

(b) Estimators from Tensor-GPST. Only non-zero feature maps are plotted.

Figure 3. Simulation Samples & Estimators from our Tensor-GPST. The feature maps are capturing signal blocks. More numerical results are available in the paper.

Real Data Application: Solar Flare Intensity Forecast

We apply our model to a solar flare intensity prediction problem where the input tensor data are 10-channel astronomical images, each having size 50×50. The results are:

Model	Training (75% of the samples)				Testing (25% of the samples)			
	RMSE	R^2	MSLL	TSS	RMSE	R^2	MSLL	TSS
Tensor-GP	0.646±0.019	0.336±0.044	1.028±0.134	0.466±0.039	0.772±0.239	0.182±0.114	1.138±0.085	0.362±0.159
CP	**0.564 ± 0.035**	**0.501 ± 0.077**	–	**0.625 ± 0.069**	0.706±0.051	0.230±0.078	–	0.398±0.092
Tucker	0.679±0.014	0.269±0.028	–	0.426±0.052	0.683±0.040	0.259±0.079	–	0.414±0.134
Tensor-GPST	0.661±0.014	0.305± 0.023	1.005±0.021	0.449±0.040	0.681±0.043	0.265±0.087	**1.035 ± 0.061**	0.412±0.112

Table 1. MSLL: Mean Standardized Log Loss; TSS: True Skill Score. Tensor-GPST achieves much better performance than classical Tensor-GP [2] and is comparable to CP/Tucker low-rank tensor regression.

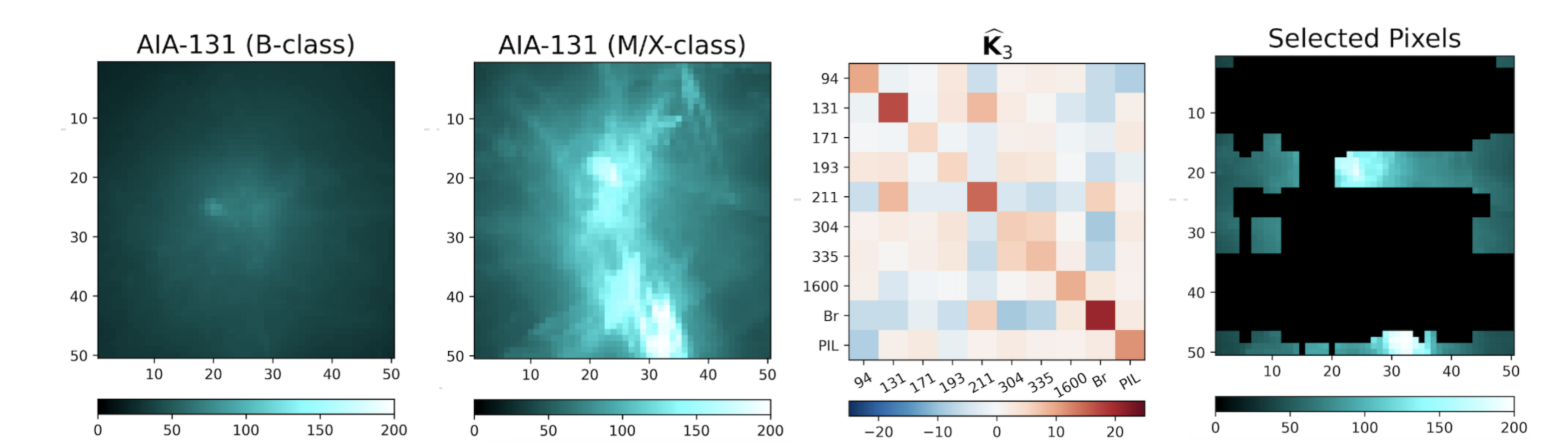


Figure 4. (Panel 1,2) AIA-131 Channel Average for B-class and M/X class flares. (Panel 3) Estimator of $\mathbf{K}_3$ reveals the channel-channel covariances. (Panel 4) Selected pixels from the tensor contraction step. Our method makes scientific interpretation for tensor regression model easier. Unit for AIA-131 is Data Number (DN) per second.

[1] Kolda, T. G., & Bader, B. W. (2009). Tensor Decompositions and Applications. SIAM review, 51(3), 455-500.
[2] Yu, R., Li, G., & Liu, Y. (2018). Tensor Regression Meets Gaussian Processes. AISTATS, PMLR, 482-490.

contact: ychenang@umich.edu