ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-6327-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICASSP49357.2023.10095485

DEEP UNFOLDED TENSOR ROBUST PCA WITH SELF-SUPERVISED LEARNING

Harry Dong,[†] *Megna Shah,*[‡] *Sean Donegan,*[‡] *Yuejie Chi*[†]

[†]Department of Electrical and Computer Engineering, Carnegie Mellon University
[‡]Materials and Manufacturing Directorate, Air Force Research Laboratory

ABSTRACT

Tensor robust principal component analysis (RPCA), which seeks to separate a low-rank tensor from its sparse corruptions, has been crucial in data science and machine learning where tensor structures are becoming more prevalent. While powerful, existing tensor RPCA algorithms can be difficult to use in practice, as their performance can be sensitive to the choice of additional hyperparameters, which are not straightforward to tune. In this paper, we describe a fast and simple self-supervised model for tensor RPCA using deep unfolding by only learning four hyperparameters. Despite its simplicity, our model expunges the need for ground truth labels while maintaining competitive or even greater performance compared to supervised deep unfolding. Furthermore, our model is capable of operating in extreme data-starved scenarios. We demonstrate these claims on a mix of synthetic data and real-world tasks, comparing performance against previously studied supervised deep unfolding methods and Bayesian optimization baselines.

Index Terms— tensors, robust principal component analysis, self-supervised learning, deep unfolding, fine tuning

1. INTRODUCTION

Becoming increasingly pervasive in data science and machine learning, tensors are powerful data structures that capture interactions between elements in higher dimensions. These structures lead to inherent properties that can be exploited for tasks that seek to analyze and process tensors. One such task is robust principal component analysis (RPCA), the problem of recovering a low-rank tensor that has been sparsely corrupted, which has found numerous applications such as surveillance, anomaly detection, and more.

Compared with its matrix counterpart, tensor RPCA faces some unique challenges, as many ideas from matrix RPCA begin to fall apart. First, blindly applying a matrix RPCA algorithm to a flattened tensor can destroy structural information [1]. Second, convex formulations, typically done with the nuclear norm as a convex relaxation to the rank constraint, is NP-hard for tensors [2]. Third, there are multiple notions of tensor decomposition and rank. As such, provable tensor RPCA algorithms have been developed for low multilinear rank [3], tubal rank [4, 5], and CP-rank [6, 7] decompositions.

In this paper, we focus on the Tucker decomposition with low multilinear rank, though we believe our approach is also extendable to other tensor decompositions. While powerful, existing efficient tensor RPCA algorithms, e.g. based on iterative scaled gradient updates as in [3], can be difficult to use in practice, as their performance is sensitive to the choice of hyperparameters, such as learning rates and thresholds. In addition, tuning these hyperparameters can be difficult, since the number of hyperparameters scales with the desired number of iterations.

Leveraging deep unfolding, the process of unrolling an iterative algorithm into a deep neural network introduced in [8], there have been considerable efforts in learning algorithmic parameters with backpropagation to improve the performance of the original algorithm. Deep unfolding for RPCA can be found in ultrasound imaging [9, 10], background subtraction [11], and the special case of positive semidefinite (PSD) low-rank matrices [12]. To generalize this idea, [13] designed a deep unfolded architecture for matrix RPCA, extendable to infinitely many RPCA iterations. However, many existing approaches [9–11, 13] train on ground truth labels which may be limited or nonexistent in practice. Though only applied to PSD low-rank matrices, [12] addressed this with an unsupervised model.

Motivated by the need to broaden the applicability and improve the performance of tensor RPCA in practice, we describe a fast and simple self-supervised model for tensor RPCA by unfolding the recently proposed algorithm in [3], due to its appealing performance. Our contributions are as follows.

1. We propose a novel self-supervised model for tensor RPCA, which can be used independently or as fine tuning for its supervised counterpart. Furthermore, it scales to an arbitrary number of RPCA iterative updates with only four learnable parameters by leveraging theoretical insights from [3].
2. Synthetic and real-world experiments on video surveillance and materials microscopy data show that our self-supervised model matches or exceeds the performance of supervised learning without the need for ground truth, especially in data-starved scenarios.

Paper organization. The rest of this paper is organized as follows. We build up the tensor RPCA algorithm from [3] in Section 2 which will be used in our learned tensor RPCA method described in Section 3. Then, we present synthetic and real-world experimental results in Section 4, followed by final remarks in Section 5.

Notation and basics of tensor algebra. Throughout this paper, we represent tensors with bold calligraphic letters (e.g. $\boldsymbol{\mathcal{X}}$) and matrices with bold capitalize letters (e.g. $\boldsymbol{X}$). For tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, $\mathcal{M}_k(\boldsymbol{\mathcal{A}})$ is the tensor matricization along the k-th mode (i.e. dimension of the tensor), $k = 1, 2, 3$, and $[\boldsymbol{\mathcal{A}}]_{i_1,i_2,i_3}$ is the (i_1, i_2, i_3)-th entry of $\boldsymbol{\mathcal{A}}$. A fiber is a vector obtained by fixing all indices except one in the tensor (e.g. $[\boldsymbol{\mathcal{A}}]_{i_1,\cdot,i_3}$). Let the inner product between two tensors $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$ be $\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \rangle = \sum_{i_1,i_2,i_3} [\boldsymbol{\mathcal{A}}]_{i_1,i_2,i_3} [\boldsymbol{\mathcal{B}}]_{i_1,i_2,i_3}$, and $\|\boldsymbol{\mathcal{A}}\|_{\mathsf{F}} = \sqrt{\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{A}} \rangle}$ and $\|\boldsymbol{\mathcal{A}}\|_1 = \sum_{i_1,i_2,i_3} |[\boldsymbol{\mathcal{A}}]_{i_1,i_2,i_3}|$ denote the Frobenius norm

The work of H. Dong and Y. Chi is supported in part by the Air Force D3OM2S Center of Excellence under FA8650-19-2-5209, by ONR under N00014-19-1-2404, by the DARPA TRIAD program under Agreement No. HR00112190099, by NSF under CCF-1901199 and ECCS-2126634, and by the Carnegie Mellon University Manufacturing Futures Initiative, made possible by the Richard King Mellon Foundation. Emails: {harryd, yuejiec}@andrew.cmu.edu, {megna.shah.1,sean.donegan}@afrl.af.mil.

and the ℓ_1-norm, respectively. Furthermore, suppose a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ has multilinear rank $\boldsymbol{r} = (r_1, r_2, r_3)$, and its Tucker decomposition is given by $\boldsymbol{\mathcal{X}} = (\boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)}, \boldsymbol{U}^{(3)}) \cdot \boldsymbol{\mathcal{G}}$ with $\boldsymbol{U}^{(k)} \in \mathbb{R}^{n_k \times r_k}$ and $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$, it follows that

$$[\boldsymbol{\mathcal{X}}]_{i_1,i_2,i_3} = \sum_{j_1,j_2,j_3} \left(\prod_{k=1,2,3} [\boldsymbol{U}^{(k)}]_{i_k,j_k} \right) [\boldsymbol{\mathcal{G}}]_{j_1,j_2,j_3}.$$

Last but not least, given *any* $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, its rank-$\boldsymbol{r}$ higher-order singular value decomposition (HOSVD) $\mathcal{H}_{\boldsymbol{r}}(\boldsymbol{\mathcal{X}})$ is given by $\mathcal{H}_{\boldsymbol{r}}(\boldsymbol{\mathcal{X}}) = (\boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)}, \boldsymbol{U}^{(3)}, \boldsymbol{\mathcal{G}})$, where $\boldsymbol{U}^{(k)} \in \mathbb{R}^{n_k \times r_k}$ contains the top r_k singular vectors of $\mathcal{M}_k(\boldsymbol{\mathcal{X}})$ and the core tensor $\boldsymbol{\mathcal{G}} = (\boldsymbol{U}^{(1)\top}, \boldsymbol{U}^{(2)\top}, \boldsymbol{U}^{(3)\top}) \cdot \boldsymbol{\mathcal{X}}$.

2. BACKGROUND ON TENSOR RPCA

2.1. Problem formulation

Suppose we observe a corrupted tensor $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ of the form

$$\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{X}}_\star + \boldsymbol{\mathcal{S}}_\star, \tag{1}$$

where $\boldsymbol{\mathcal{X}}_\star$ is a low-rank tensor with the multilinear rank $\boldsymbol{r} = (r_1, r_2, r_3)$, whose Tucker decomposition is given by $\boldsymbol{\mathcal{X}}_\star = (\boldsymbol{U}_\star^{(1)}, \boldsymbol{U}_\star^{(2)}, \boldsymbol{U}_\star^{(3)}) \cdot \boldsymbol{\mathcal{G}}_\star$ with $\boldsymbol{U}_\star^{(k)} \in \mathbb{R}^{n_k \times r_k}$ for $k = 1, 2, 3$, and $\boldsymbol{\mathcal{G}}_\star \in \mathbb{R}^{r_1 \times r_2 \times r_3}$. Moreover, $\boldsymbol{\mathcal{S}}_\star$ is an α-sparse corruption tensor, i.e. $\boldsymbol{\mathcal{S}}_\star$ contains at most $0 \leq \alpha < 1$ fraction nonzero values along each fiber. Given $\boldsymbol{\mathcal{Y}}$ and $\boldsymbol{r}$, the goal of tensor RPCA aims to accurately obtain $\boldsymbol{\mathcal{X}}_\star$ and $\boldsymbol{\mathcal{S}}_\star$.

2.2. Tensor RPCA via ScaledGD

Recently, Dong et al. [3] proposed a fast and scalable method for tensor RPCA with provable performance guarantees. Specifically, they try to minimize the objective function

$$\mathcal{L}(\boldsymbol{F}, \boldsymbol{\mathcal{S}}) \coloneqq \frac{1}{2} \left\| (\boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)}, \boldsymbol{U}^{(3)}) \cdot \boldsymbol{\mathcal{G}} + \boldsymbol{\mathcal{S}} - \boldsymbol{\mathcal{Y}} \right\|_{\mathsf{F}}^2, \tag{2}$$

where $\boldsymbol{F} = (\boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)}, \boldsymbol{U}^{(3)}, \boldsymbol{\mathcal{G}})$ and $\boldsymbol{\mathcal{S}}$ are the estimates of the factors of the low-rank tensor and the sparse tensor, respectively. The algorithm proceeds by iterative updates of the factors $\boldsymbol{F}$ via scaled gradient descent (ScaledGD) [14–17], followed by filtering the outliers $\boldsymbol{\mathcal{S}}$ using the shrinkage operator $\mathcal{T}_\zeta(\cdot)$, defined as

$$[\mathcal{T}_\zeta(\boldsymbol{\mathcal{X}})]_{i,j,k} \coloneqq \operatorname{sgn}([\boldsymbol{\mathcal{X}}]_{i,j,k}) \cdot \max(0, |[\boldsymbol{\mathcal{X}}]_{i,j,k}| - \zeta). \tag{3}$$

The details of the proposed ScaledGD algorithm are summarized in Algorithm 1. Beginning with a careful initialization using the spectral method, given by

$$\left(\boldsymbol{U}_0^{(1)}, \boldsymbol{U}_0^{(2)}, \boldsymbol{U}_0^{(3)}, \boldsymbol{\mathcal{G}}_0\right) = \mathcal{H}_{\boldsymbol{r}}\left(\boldsymbol{\mathcal{Y}} - \mathcal{T}_{\zeta_0}(\boldsymbol{\mathcal{Y}})\right), \tag{4}$$

for each iteration, ScaledGD makes the following updates:

$$\boldsymbol{\mathcal{S}}_{t+1} = \mathcal{T}_{\zeta_{t+1}}\left(\boldsymbol{\mathcal{Y}} - (\boldsymbol{U}_t^{(1)}, \boldsymbol{U}_t^{(2)}, \boldsymbol{U}_t^{(3)}) \cdot \boldsymbol{\mathcal{G}}_t\right), \tag{5a}$$

$$\boldsymbol{U}_{t+1}^{(k)} = \boldsymbol{U}_t^{(k)} - \eta \nabla_{\boldsymbol{U}_t^{(k)}} \mathcal{L}(\boldsymbol{F}_t, \boldsymbol{\mathcal{S}}_{t+1}) \left(\breve{\boldsymbol{U}}_t^{(k)\top} \breve{\boldsymbol{U}}_t^{(k)}\right)^{-1} \tag{5b}$$

for $k = 1, 2, 3$, and

$$\boldsymbol{\mathcal{G}}_{t+1} = \boldsymbol{\mathcal{G}}_t - \eta \left(\tilde{\boldsymbol{U}}_t^{(1)}, \tilde{\boldsymbol{U}}_t^{(2)}, \tilde{\boldsymbol{U}}_t^{(3)}\right) \cdot \nabla_{\boldsymbol{\mathcal{G}}_t} \mathcal{L}(\boldsymbol{F}_t, \boldsymbol{\mathcal{S}}_{t+1}), \tag{5c}$$

Algorithm 1 ScaledGD for 3rd order tensor RPCA

Input: the observed tensor $\boldsymbol{\mathcal{Y}}$, the multilinear rank $\boldsymbol{r}$, learning rate η, and threshold schedule $\{\zeta_t\}_{t=0}^T$.
Init: $\left(\boldsymbol{U}_0^{(1)}, \boldsymbol{U}_0^{(2)}, \boldsymbol{U}_0^{(3)}, \boldsymbol{\mathcal{G}}_0\right) = \mathcal{H}_{\boldsymbol{r}}\left(\boldsymbol{\mathcal{Y}} - \mathcal{T}_{\zeta_0}(\boldsymbol{\mathcal{Y}})\right)$.
for $t = 0, 1, \ldots, T-1$ **do**
  Update $\boldsymbol{\mathcal{S}}_{t+1}$ via (5a);
  Update $\boldsymbol{F}_{t+1} = \left(\boldsymbol{U}_{t+1}^{(1)}, \boldsymbol{U}_{t+1}^{(2)}, \boldsymbol{U}_{t+1}^{(3)}, \boldsymbol{\mathcal{G}}_{t+1}\right)$ via (5b) and (5c);
end for
Output: $\boldsymbol{F}_T = \left(\boldsymbol{U}_T^{(1)}, \boldsymbol{U}_T^{(2)}, \boldsymbol{U}_T^{(3)}, \boldsymbol{\mathcal{G}}_T\right)$.

where $\eta > 0$ is the learning rate, and $\{\zeta_t\}_{t=0}^T$ is the threshold schedule up to T iterations. Here,

$$\breve{\boldsymbol{U}}_t^{(1)} = (\boldsymbol{U}_t^{(3)} \otimes \boldsymbol{U}_t^{(2)}) \mathcal{M}_1(\boldsymbol{\mathcal{G}}_t)^\top,$$
$$\breve{\boldsymbol{U}}_t^{(2)} = (\boldsymbol{U}_t^{(3)} \otimes \boldsymbol{U}_t^{(1)}) \mathcal{M}_2(\boldsymbol{\mathcal{G}}_t)^\top,$$
$$\breve{\boldsymbol{U}}_t^{(3)} = (\boldsymbol{U}_t^{(2)} \otimes \boldsymbol{U}_t^{(1)}) \mathcal{M}_3(\boldsymbol{\mathcal{G}}_t)^\top,$$

and $\tilde{\boldsymbol{U}}_t^{(k)} = \left(\boldsymbol{U}_t^{(k)\top} \boldsymbol{U}_t^{(k)}\right)^{-1}$, $k = 1, 2, 3$, where $\otimes$ denotes the Kronecker product. In [3], it is shown ScaledGD perfectly recovers the low-rank tensor $\boldsymbol{\mathcal{X}}_\star$ and the sparse tensor $\boldsymbol{\mathcal{S}}_\star$ under mild assumptions as long as the low-rank tensor is incoherent and the corruption level α is not too large. Moreover, ScaledGD converges at a linear rate independent of the condition number of $\boldsymbol{\mathcal{X}}_\star$ [3], making it more appealing than the vanilla gradient descent approach that is much more sensitive to ill-conditioning. However, translating the theoretical advantage into practice requires carefully tuned hyperparamters, η and $\{\zeta_t\}_{t=0}^T$, which unfortunately are not readily available.

3. PROPOSED METHOD

We aim to greatly broaden the applicability of Algorithm 1 with learned hyperparameters by leveraging self-supervised learning (SSL) under a deep-unfolding perspective [8] of Algorithm 1.

3.1. Unrolling-aided hyperparameter tuning

Based on the insights from theoretical analysis [3, Theorem 1], we use a threshold schedule that decays at every iteration to reduce the number of hyperparameters. In other words,

$$\zeta_{t+1} = \rho \zeta_t$$

for $t \geq 1$ and $0 < \rho < 1$, so there are only 4 hyperparameters that need tuning: ζ_0, ζ_1, ρ, and η. By unrolling Algorithm 1, we obtain an unfolded version of Algorithm 1, where ζ_0, ζ_1, ρ, and η are learnable; see Fig. 1 for more details. The model consists of a feed-forward layer and a recurrent layer to model the initialization and iterative steps of Algorithm 1.

It remains to specify the loss function used to learn the hyperparameters. One immediate idea is to optimize the performance in reconstructing the true tensor $\boldsymbol{\mathcal{X}}_\star$ by minimizing

$$\mathcal{L}_{\text{SL}}(\boldsymbol{F}) \coloneqq \frac{\left\| \boldsymbol{\mathcal{X}}_\star - (\boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)}, \boldsymbol{U}^{(3)}) \cdot \boldsymbol{\mathcal{G}} \right\|_{\mathsf{F}}^2}{\|\boldsymbol{\mathcal{X}}_\star\|_{\mathsf{F}}^2}, \tag{6}$$

which we refer to as supervised learning (SL) similar to [13]. While this is possible when trained on synthetic data—where we have access to the ground truth—it may not be applicable in general in the absence of ground truth.

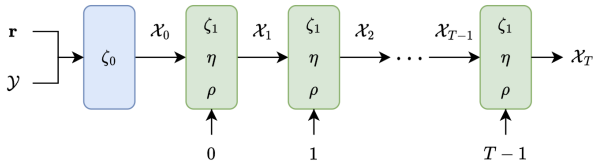


Fig. 1: Network architecture. $\mathcal{Y}$ and $\boldsymbol{r}$ are fed into the first layer, the initialization step in Algorithm 1. Subsequent recurrent layers follow Algorithm 1 to update $\mathcal{X}_t = (\boldsymbol{U}_t^{(1)}, \boldsymbol{U}_t^{(2)}, \boldsymbol{U}_t^{(3)}) \cdot \mathcal{G}_t$. Recurrent layers require the iteration count t due to our definition of the $(t+1)$-th thresholding parameter, $\zeta_{t+1} := \zeta_1 \rho^t$ for $t \geq 0$. We use softplus activations to ensure $\zeta_0, \zeta_1, \eta > 0$ and the sigmoid function to make $0 < \rho < 1$.

One might naturally wonder if (2) can be used to optimize the hyperparameters. Unfortunately, with the decaying threshold schedule, (2) can be trivially solved as we increase the number of iterations T or choosing ρ close to 0, since the threshold ζ_t will rapidly approach 0, making $\mathcal{S}_t \approx \mathcal{Y} - \mathcal{X}_t$ by (5a). Hence, the objective function (2) is no longer appropriate in learning the hyperparameters. As such, we will instead use the following loss function:

$$\mathcal{L}_{\text{SSL}}(\boldsymbol{F}) := \frac{1}{\|\mathcal{Y}\|_{\text{F}}^2} \left\| \mathcal{Y} - \left(\boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)}, \boldsymbol{U}^{(3)}\right) \cdot \mathcal{G} \right\|_1 \tag{7}$$

to encourage sparsity of the corruption tensor [12, 18]. We refer to (7) as the self-supervised learning (SSL) loss since it does not require the ground truth and therefore, more amenable to real-data scenarios.

3.2. From supervised learning to self-supervised learning

Depending on the loss function, our model can be used for both supervised or self-supervised learning of the hyperparameters. For supervised learning, suppose we have access to $\mathcal{X}_\star$ for all tensor RPCA problem instances in a training dataset, then the hyperparameters can be tuned by optimizing (6).

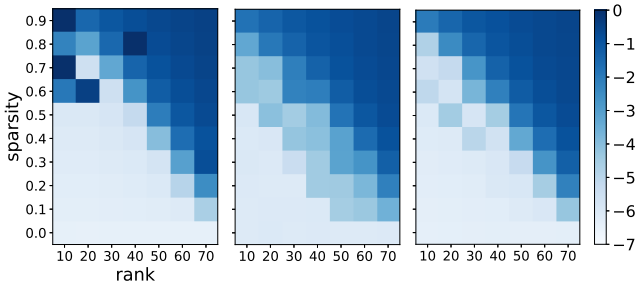
A drawback of a purely supervised approach is the assumption of a "one size fits all" set of hyperparameters that can be applied to other similar tensors. In fact, tensor inputs into RPCA are exceptionally complex and in general, cannot be adequately described in a few features such as their sparsity levels, ranks, and condition numbers. Luckily, tuned hyperparameters obtained from training serve as warm starts for further refinements. Similar to [12], we first learn a fixed set of hyperparameters on the training data (when available), which will be fine tuned during test time by minimizing (7) individually for each tensor, leading to a self-supervised learning paradigm.

4. EXPERIMENTS

We conduct synthetic and real-world data experiments to corroborate the effectiveness of the proposed learned tensor RPCA approach. Our code is available at https://github.com/hdong920/Tensor_RPCA_ScaledGD.

4.1. Synthetic data

We explore the effect of fine tuning with SSL for each combination of $\alpha \in \{0, 0.1, \ldots, 0.9\}$ and $r \in \{10, 20, \ldots, 70\}$ with fixed $n = 100$. When a rank (r, r, r) tensor $\mathcal{Y} \in \mathbb{R}^{n \times n \times n}$ needs to be sampled, we first randomly generate factor matrices $\boldsymbol{U}_\star^{(1)}, \boldsymbol{U}_\star^{(2)}, \boldsymbol{U}_\star^{(3)} \in \mathbb{R}^{n \times r}$ with orthonormal columns and core tensor $\mathcal{G}_\star$ where $[\mathcal{G}_\star]_{i,i,i} = \kappa^{-(i-1)/(r-1)}$ for $\kappa = 5$ and 0 elsewhere



(a) Baseline (b) Supervised (c) Fine tuning

Fig. 2: The log mean relative recovery error, $\frac{\|\mathcal{X}_\star - \mathcal{X}_T\|_{\text{F}}}{\|\mathcal{X}_\star\|_{\text{F}}}$ for each combination of α and r of Algorithm 1 using hyperparameters found by the baseline, supervised learning, and supervised learning with self-supervised fine tuning.

to construct $\mathcal{X}_\star = (\boldsymbol{U}_\star^{(1)}, \boldsymbol{U}_\star^{(2)}, \boldsymbol{U}_\star^{(3)}) \cdot \mathcal{G}_\star$. In addition, $\mathcal{S}_\star$ is a sparse tensor with α-fraction of its entries drawn from a uniform distribution in $(-\theta, \theta)$, where $\theta := \frac{1}{n^3} \|\mathcal{X}_\star\|_1$. All experiments in this section were run for $T = 100$ iterations of Algorithm 1.

First, we perform supervised learning by minimizing (6) similar to [13] for each combination of α and r. For each gradient step, we generate a new sample (a tensor RPCA problem instance). We train for 1000 gradient steps with a decaying learning rate scheduler. Next, we perform SSL for 500 gradient steps to fine tune the hyperparameters for 20 samples generated for each combination of α and r by minimizing (7). As the baseline, we use Optuna [19], a Bayesian optimization-based hyperparameter tuning package, to minimize (7) individually for 20 samples. We run Optuna for 500 iterations for each sample over a search space containing the learned hyperparameters. Fig. 2 shows that the baseline accurately recovers $\mathcal{X}_\star$ across a wide range of cases, though it can be unstable in low rank and high sparsity scenarios. Supervised learning improves the stability of these scenarios but performs slightly worse along the edge of the phase transition. Although fine tuning adds more computation, we simultaneously obtain more stable results than the baseline and gain a significant improvement on recovering $\mathcal{X}_\star$ over supervised learning, especially along the edge of the phase transition.

During self-supervised fine tuning, we track the percent change of the RPCA hyperparameters' fined tuned values relative to their final values from supervised learning. Table 1 contains the quartiles of these percent changes for samples that observed a more than a 90% reduction in recovery error after fine tuning. We see that RPCA hyperparameters can be quite sensitive where tuning by grid search would require very fine sampling. Fine tuning adapts to this sensitivity which supervised learning alone cannot address.

Quartile	ζ_0	ζ_1	η	ρ
Q1	0.93%	3.06%	-14.03%	3.36%
Q2	5.17%	28.91%	-3.38%	6.20%
Q3	15.73%	36.56%	7.85%	6.68%

Table 1: Fine tuning parameter percent change quartiles for samples with greater than 90% reduction in recovery loss after fine tuning.

4.2. Background subtraction in video surveillance

Next, we apply learned tensor RPCA from scratch on the background subtraction task using the Background Models Challenge real videos dataset [20], which contains 9 videos of varying shapes. We use the first 6 to train two separate models via supervised learn-

(a) Supervised learning

(b) Self-supervised learning

Fig. 3: Background subtraction of a video containing a moving train and worker. From left to right, the columns are the input video, extracted sparse foreground, and low-rank background.

ing and SSL with $T = 150$ for 15 epochs, which are tested on the last 3 videos. Models assumed rank 1 along the time dimension and full rank for all others. For SSL, it minimizes (7). However, for supervised learning, as this dataset includes only ground truth binary foreground masks (Boolean labels that indicate if a pixel is part of the foreground), we cannot use (6). Defining $\boldsymbol{\mathcal{M}}$ to be the binary foreground mask of the same shape as $\boldsymbol{\mathcal{Y}}$, we know $\boldsymbol{\mathcal{S}}_\star$ and $\boldsymbol{\mathcal{M}}$ share the same support. Letting $\odot$ be the Hadamard product, we use

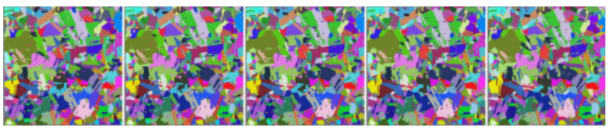
$$\mathcal{L}_{\text{SM}}(\boldsymbol{F}) := \frac{\left\| \left(\boldsymbol{\mathcal{Y}} - \left(\boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)}, \boldsymbol{U}^{(3)} \right) \cdot \boldsymbol{\mathcal{G}} \right) \odot (1 - \boldsymbol{\mathcal{M}}) \right\|_{\text{F}}^2}{\| \boldsymbol{\mathcal{Y}} \odot (1 - \boldsymbol{\mathcal{M}}) \|_{\text{F}}^2}$$

for supervised learning since $\boldsymbol{\mathcal{Y}} \odot (1 - \boldsymbol{\mathcal{M}}) = \boldsymbol{\mathcal{X}}_\star \odot (1 - \boldsymbol{\mathcal{M}})$. Because we only assume low rank along the time dimension, iterative updates of Algorithm 1 are skipped for other modes to save computation. Fig. 3 display no significant visual difference between results from the two models, suggesting the benefit of SSL in alleviating the need of labeled training data for this task. This is backed up by their similar masked average relative recovery test errors, $\sqrt{\mathcal{L}_{\text{SM}}(\boldsymbol{F}_T)}$. We observe that as the assumed time dimension rank increased, the more fine tuning improved relative recovery error.

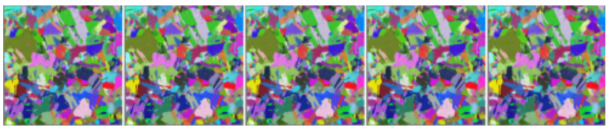
4.3. Electron backscatter diffraction microscopy

We compare our SSL-based tensor RPCA method and the baseline (tuned by Optuna [19] using the loss (7)) on high-dimensional microscopy data from [21]. To provide context, data collection involved iteratively performing electron backscatter diffraction (EBSD) on a nickel-based superalloy. EBSD collects an image where the orientation of the local crystal, represented as Euler angles, is stored on each pixel. The superalloy was mechanically polished after each EBSD image collection, removing approximately 1 micron of material. This was repeated many times to obtain a sequence of slices containing orientation measurements, with an in-plane pixel spacing of 1 micron. The result is a volume of orientation information that can be considered as a high-dimensional tensor. See Fig. 4a for a visualization of consecutive slices. Note this dataset has no inherent notion of ground truth that we can perform supervised learning on for RPCA and contains only one sample.

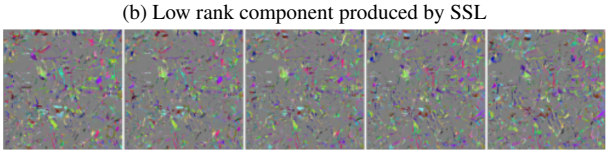
The motivation of applying RPCA here is that materials scientists usually arduously parse through these these slices manually. Based on Fig. 4a, changes from slice to slice are very subtle. Being able to separate the large slow moving sections from the sparse rapidly evolving parts could allow materials scientists to quickly identify critical rare events with implications on life-limiting behavior. Due to the dataset's significant amount of structure, we train our model for only 10 iterations of hyperparameter updates on $(250 \times 250 \times 250 \times 3)$ samples where we identify unique parameters for each sample using $\boldsymbol{r} = (250, 25, 250, 3)$ and $T = 10$, again skipping updates along full rank modes. From Fig. 4b and 4c, our method extracts sparse components that capture detailed changes across slices and low-rank components that contain macro-scale patterns. We repeat the experiment for 50 iterations of the baseline method, with the results shown in Fig. 4d and 4e. Although the baseline resulted in similar loss values and captured similar quality low rank and sparse structures, it took many more iterations (about 5 times more) to obtain these results. Furthermore, we note that the per-iteration hyperparameter update runtimes of the baseline and our model were very similar, so our model also ran about 5 times faster.



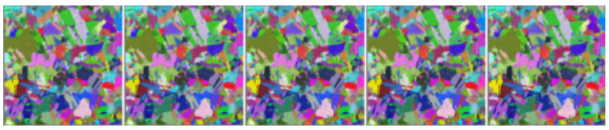
(a) Original slices



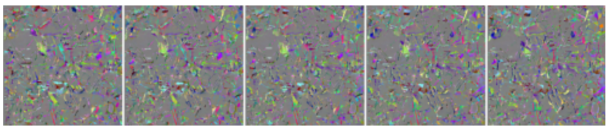
(b) Low rank component produced by SSL



(c) Sparse component produced by SSL



(d) Low rank component produced by Baseline



(e) Sparse component produced by Baseline

Fig. 4: The tensor RPCA outputs on example EBSD slices.

5. CONCLUSIONS

We have illustrated and empirically shown two ways deep unfolded SSL can be used to improve current approaches to tensor RPCA. First, a deep unfolded version of Algorithm 1 with a new ℓ_1-based loss function to train the hyperparameters can perform equivalently with supervised models and quickly provide satisfactory results in tasks where supervision is inapplicable. Second, self-supervised fine tuning improves recovery by finding a unique set of RPCA hyperparameters for each RPCA problem instance. Altogether, we have simultaneously extended the applicability of tensor RPCA to data-starved settings and enhanced its performance. Possible future adaptations include extending our method to design similar models for tensor algorithms that are centered around reconstruction like tensor completion. It would also be of interest to replace our recurrent layers with other architectures for sequential inputs like transformers.

6. REFERENCES

[1] M. Yuan and C.-H. Zhang, "On tensor completion via nuclear norm minimization," *Foundations of Computational Mathematics*, vol. 16, no. 4, pp. 1031–1068, 2016.

[2] S. Friedland and L.-H. Lim, "Nuclear norm of higher-order tensors," *Mathematics of Computation*, vol. 87, no. 311, pp. 1255–1281, 2018.

[3] H. Dong, T. Tong, C. Ma, and Y. Chi, "Fast and provable tensor robust principal component analysis via scaled gradient descent," *arXiv preprint arXiv:2206.09109*, 2022.

[4] C. Lu, J. Feng, Y. Chen, W. Liu, Z. Lin, and S. Yan, "Tensor robust principal component analysis: Exact recovery of corrupted low-rank tensors via convex optimization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5249–5257.

[5] ——, "Tensor robust principal component analysis with a new tensor nuclear norm," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 4, pp. 925–938, 2019.

[6] D. Driggs, S. Becker, and J. Boyd-Graber, "Tensor robust principal component analysis: Better recovery with atomic norm regularization," *arXiv preprint arXiv:1901.10991*, 2019.

[7] A. Anandkumar, P. Jain, Y. Shi, and U. N. Niranjan, "Tensor vs. matrix methods: Robust tensor decomposition under block sparse perturbations," in *Artificial Intelligence and Statistics*. PMLR, 2016, pp. 268–276.

[8] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proceedings of the 27th international conference on international conference on machine learning*, 2010, pp. 399–406.

[9] O. Solomon, R. Cohen, Y. Zhang, Y. Yang, Q. He, J. Luo, R. J. van Sloun, and Y. C. Eldar, "Deep unfolded robust PCA with application to clutter suppression in ultrasound," *IEEE transactions on medical imaging*, vol. 39, no. 4, pp. 1051–1063, 2019.

[10] R. Cohen, Y. Zhang, O. Solomon, D. Toberman, L. Taieb, R. J. van Sloun, and Y. C. Eldar, "Deep convolutional robust PCA with application to ultrasound imaging," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3212–3216.

[11] H. Van Luong, B. Joukovsky, Y. C. Eldar, and N. Deligiannis, "A deep-unfolded reference-based RPCA network for video foreground-background separation," in *2020 28th European Signal Processing Conference (EUSIPCO)*. IEEE, 2021, pp. 1432–1436.

[12] C. Herrera, F. Krach, A. Kratsios, P. Ruyssen, and J. Teichmann, "Denise: Deep learning based robust PCA for positive semidefinite matrices," *stat*, vol. 1050, p. 5, 2020.

[13] H. Cai, J. Liu, and W. Yin, "Learned robust PCA: A scalable deep unfolding approach for high-dimensional outlier detection," *Advances in Neural Information Processing Systems*, vol. 34, pp. 16 977–16 989, 2021.

[14] T. Tong, C. Ma, and Y. Chi, "Accelerating ill-conditioned low-rank matrix estimation via scaled gradient descent," *Journal of Machine Learning Research*, vol. 22, pp. 1–63, 2021.

[15] T. Tong, C. Ma, A. Prater-Bennette, E. Tripp, and Y. Chi, "Scaling and scalability: Provable nonconvex low-rank tensor estimation from incomplete measurements," *Journal of Machine Learning Research*, vol. 23, no. 163, pp. 1–77, 2022.

[16] T. Tong, C. Ma, and Y. Chi, "Accelerating ill-conditioned robust low-rank tensor regression," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 9072–9076.

[17] ——, "Low-rank matrix recovery with scaled subgradient methods: Fast and robust convergence without the condition number," *IEEE Transactions on Signal Processing*, vol. 69, pp. 2396–2409, 2021.

[18] V. Charisopoulos, Y. Chen, D. Davis, M. Díaz, L. Ding, and D. Drusvyatskiy, "Low-rank matrix recovery with composite optimization: good conditioning and rapid convergence," *Foundations of Computational Mathematics*, vol. 21, no. 6, pp. 1505–1593, 2021.

[19] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[20] A. Vacavant, T. Chateau, A. Wilhelm, and L. Lequievre, "A benchmark dataset for outdoor foreground/background extraction," in *Asian Conference on Computer Vision*. Springer, 2012, pp. 291–300.

[21] M. G. Chapman, M. N. Shah, S. P. Donegan, J. M. Scott, P. A. Shade, D. Menasche, and M. D. Uchic, "Afrl additive manufacturing modeling series: challenge 4, 3d reconstruction of an in625 high-energy diffraction microscopy sample using multimodal serial sectioning," *Integrating Materials and Manufacturing Innovation*, vol. 10, no. 2, pp. 129–141, 2021.