A Proof of theorem

Proof. We provide the proof below (first by submultiplicativity),

$$\begin{aligned}
\left\|\mathbf{u}^{t+1}-\mathbf{u}^{*}\right\|_2 &= \left\|\frac{\beta \mathbf{v}_2}{2\|\mathbf{u}^{*}\|_2}\left(\mathbf{I}-\frac{\mathbf{u}^{*\top}\mathbf{u}^{*}}{\|\mathbf{u}^{*}\|_2^2}\right)\mathbf{V}_3-\frac{\beta \mathbf{v}_2}{2\|\mathbf{u}^{t}\|_2}\left(\mathbf{I}-\frac{\mathbf{u}^{t\top}\mathbf{u}^{t}}{\|\mathbf{u}^{t}\|_2^2}\right)\mathbf{V}_3\right\|_2 \\
&\leq \left\|\frac{\beta \mathbf{v}_2}{2}\right\|_2 \|\mathbf{V}_3\|_F \left\|\frac{1}{\|\mathbf{u}^{*}\|_2}\left(\mathbf{I}-\frac{\mathbf{u}^{*\top}\mathbf{u}^{*}}{\|\mathbf{u}^{*}\|_2^2}\right)-\frac{1}{\|\mathbf{u}^{t}\|_2}\left(\mathbf{I}-\frac{\mathbf{u}^{t\top}\mathbf{u}^{t}}{\|\mathbf{u}^{t}\|_2^2}\right)\right\|_2 \\
&= \left\|\frac{\beta \mathbf{v}_2}{2}\right\|_2 \|\mathbf{V}_3\|_F \left\|\left(\frac{1}{\|\mathbf{u}^{*}\|_2}-\frac{1}{\|\mathbf{u}^{t}\|_2}\right)+\left(\frac{\mathbf{u}^{t\top}\mathbf{u}^{t}}{\|\mathbf{u}^{t}\|_2^3}-\frac{\mathbf{u}^{*\top}\mathbf{u}^{*}}{\|\mathbf{u}^{*}\|_2^3}\right)\right\|_2 .
\end{aligned}$$

Then, we decompose the last term into two parts and have

$$\begin{aligned}
\frac{1}{\|\mathbf{u}^{*}\|_2}-\frac{1}{\|\mathbf{u}^{t}\|_2} &= \frac{\|\mathbf{u}^{t}\|_2-\|\mathbf{u}^{*}\|_2}{\|\mathbf{u}^{*}\|_2\|\mathbf{u}^{t}\|_2} \leq \frac{\|\mathbf{u}^{t}-\mathbf{u}^{*}\|_2}{\|\mathbf{u}^{*}\|_2\|\mathbf{u}^{t}\|_2} \leq \frac{\|\mathbf{u}^{t}-\mathbf{u}^{*}\|_2}{m^2}, \\
\frac{\mathbf{u}^{t\top}\mathbf{u}^{t}}{\|\mathbf{u}^{t}\|_2^3}-\frac{\mathbf{u}^{*\top}\mathbf{u}^{*}}{\|\mathbf{u}^{*}\|_2^3} &= \left(\frac{\mathbf{u}^{t\top}\mathbf{u}^{t}}{\|\mathbf{u}^{t}\|_2^3}-\frac{\mathbf{u}^{*\top}\mathbf{u}^{t}}{\|\mathbf{u}^{t}\|_2^3}\right)+\left(\frac{\mathbf{u}^{*\top}\mathbf{u}^{t}}{\|\mathbf{u}^{t}\|_2^3}-\frac{\mathbf{u}^{*\top}\mathbf{u}^{*}}{\|\mathbf{u}^{t}\|_2^3}\right)+\left(\frac{\mathbf{u}^{*\top}\mathbf{u}^{*}}{\|\mathbf{u}^{t}\|_2^3}-\frac{\mathbf{u}^{*\top}\mathbf{u}^{*}}{\|\mathbf{u}^{*}\|_2^3}\right) \\
&= \frac{(\mathbf{u}^{t\top}-\mathbf{u}^{*\top})\mathbf{u}^{t}}{\|\mathbf{u}^{t}\|_2^3}+\frac{\mathbf{u}^{*\top}(\mathbf{u}^{t}-\mathbf{u}^{*})}{\|\mathbf{u}^{t}\|_2^3}+\frac{\|\mathbf{u}^{*}\|_2^2(\|\mathbf{u}^{*}\|_2^3-\|\mathbf{u}^{t}\|_2^3)}{\|\mathbf{u}^{t}\|_2^3\|\mathbf{u}^{*}\|_2^3} \\
&\leq \frac{\|\mathbf{u}^{t}-\mathbf{u}^{*}\|_2}{m^2}+\frac{M\|\mathbf{u}^{t}-\mathbf{u}^{*}\|_2}{m^3}+\frac{(M+2m)\|\mathbf{u}^{t}-\mathbf{u}^{*}\|_2}{m^3}.
\end{aligned}$$

The proof is complete by triangular inequality. □

B Analysis of the Iterative Rule

In this section, we study how many rounds of the iterative rules are needed to achieve a good classification result. This study is conducted on HAR and Sleep-EDF with five random seeds.

Linear Convergence Speed. First, we study the convergence speed (when $\beta = 2$). We consider two scenarios: (Scenario 1) when the bases $\{\mathbf{A}, \mathbf{B}, \mathbf{C}\}$ are initialized as random matrices; (Scenario 2) when the based $\{\mathbf{A}, \mathbf{B}, \mathbf{C}\}$ are already learned. We use the average relative difference (of the F-norm) as the convergence measure, i.e., $\frac{1}{N}\sum_{k=1}^{N}\frac{\|\mathbf{x}_{t+1}^{(k)}-\mathbf{x}_{t}^{(k)}\|}{\|\mathbf{x}_{t}^{(k)}\|} = \frac{1}{N}\sum_{k=1}^{N}\frac{\|\mathbf{x}_{\text{impr}}^{(k)}-\mathbf{x}_{\text{init}}^{(k)}\|}{\|\mathbf{x}_{\text{init}}^{(k)}\|}$, where $\mathbf{x}^{(k)}$ means the k-th row of $\mathbf{X}$ and t means the number of rounds of the iterative rule. We test on $t = 1, 2, 3, 4, 8$. The comparison is shown in Figure 3. Both scenarios verify the linear convergence speed.

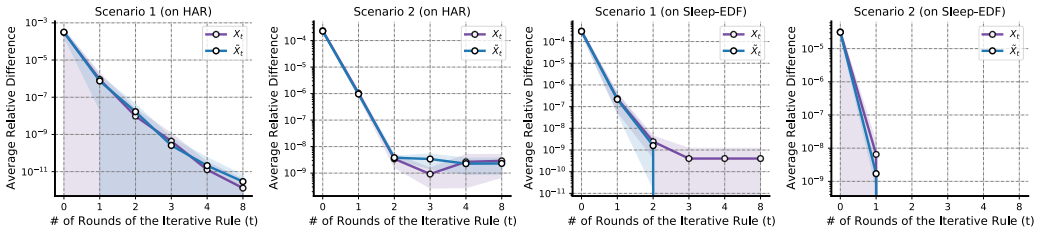

Figure 3: Verification of Convergence Speed. When the # of iteration t becomes larger, the average relative difference can exceed the minimum precision and go to zero (like Scenario 2 on Sleep-EDF).

One Round is Enough. We consider the performance of downstream classification with different rounds of iterative rules. The results are shown in Table 3 and Table 4.

Table 3: Performance with Different Rounds (on HAR)

# of rounds (t)	1	2	3	4	8
time per sweep	8.774s	10.316s	11.224s	12.781s	17.402s
accuracy (%)	93.30 ± 0.413	93.35 ± 0.172	93.35 ± 0.150	93.35 ± 0.122	93.35 ± 0.122

Table 4: Performance with Different Rounds (on Sleep-EDF)

# of rounds (t)	1	2	3	4	8
time per sweep	148.375s	160.924s	173.361s	188.193s	242.386s
accuracy (%)	85.25 $\pm$ 0.209	85.33 $\pm$ 0.173	85.31 $\pm$ 0.178	85.31 $\pm$ 0.177	85.31 $\pm$ 0.177

We observe that with an increasing number of rounds of iterative rule, the classification results will not improve further, however, the time consumption increases. Thus, we use only one round of the iterative rule in our experiments.

C Experimental Details

C.1 Dataset Processing

Sleep-EDF is publicly available, which contains 153 full-night EEG (from Fpz-Cz and Pz-Oz electrode locations), EOG (horizontal), and submental chin EMG recordings, under Open Data Commons Attribution License v1.0 and *MGH Sleep* is provided by (Biswal et al., 2018), where F3-M2, F4-M1, C3-M2, C4-M1, O1-M2, O2-M1 channels are used, containing 6,478 recordings. These two EEG datasets are processed in a similar way. First, the raw data are (long) recordings of each subject. On subject-level, these recordings are categorized into unlabeled and labeled sets by $90\% : 10\%$. Then the labeled sets are further separated into training and test by $5\% : 5\%$. Next, within each set (unlabeled, training, test), recordings are further segmented into disjoint 30-second-long periods, which are the data samples in our study. Each data sample is represented as a matrix, *channel by timestamp*, and they are associated with one of five sleep stages, Awake (W), Non-REM stage 1 (N1), Non-REM stage 2 (N2), Non-REM stage 3 (N3), and REM stage (R). *HAR* is also public, collected as 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz by the embedded accelerometer and gyroscope. It has been randomly partitioned into 70% and 30%. We use 70% as the unlabeled data (labels removed) and split the other part: 15% as training data and 15% as the test data. The license of this dataset is included in their citation. *PTB-XL* is a public ECG dataset, which contains 21,837 clinical 12-lead ECGs (male: 11,379 and female: 10,458) of 10-second length with a sampling frequency of 500 Hz. We randomly split the dataset into unlabeled (remove the labels) and labeled sets by $90\% : 10\%$; then the labeled sets are further separated into training and test by $5\% : 5\%$. This dataset is under Open BSD 3.0.

All datasets are de-identified (e.g., no names, no locations), and there is no offensive content. All the labels are also provided along with the datasets. The label distributions are shown in Table 5.

Table 5: Class Label Distribution

Name	Label Distribution
Sleep-EDF	W: 68.8%, N1: 5.2%, N2: 16.6%, N3: 3.2%, R: 6.2%
HAR	Walk: 16.72%, Walk upstairs: 14.99%, Walk downstairs: 13.65%, Sit: 17.25%, Stand: 18.51%, Lay: 18.88%
PTB-XL	Male: 52.11%, Female: 47.89%
MGH Sleep	W: 44.3%, N1: 9.9%, N2: 14.4%, N3: 17.6%, R: 13.8%

All data processing and model implementations largely follow Yang et al. (2021b)[3].

C.2 STFT Transform

We find that directly using the raw data (spatial information) does not provide good results, even for the deep learning models. Thus, we take Short-Time Fourier Transforms (STFT) as a preprocessing step. From a single channel, we can extract both the amplitude and phase information, which is then stacked together as two different channels. After STFT, each data sample becomes a three-order tensor, *channel by frequency by timestamp*. The FFT size is 256 and hop length is 32 for Sleep-EDF; the FFT size is 64 and hop length is 2 for HAR; the FFT size is 256 and hop length is 64 for PTB-XL;

[3] https://github.com/ycq091044/ContraWR

and the FFT size is 512 and hop length is 128 for MGH. We use these third-order tensors as final input data samples for all models.

C.3 Data Augmentation

In our work, we build our feature extractor $\mathbf{f}(\cdot)$ from tensor decomposition tool, which may not be as expressive/flexible as deep neural networks in SSL, and thus we also ask the augmentation methods to allow a similar component-based representation for the perturbed data. Use EEG signals as an example, we consider jittering and bandpass filtering as two augmentation methods in the experiments, which perturb the signal frequency information and will not significantly change the low-rank structure of the data.

As mentioned in the main text, we consider three different augmentation methods: (i) *Jittering* adds additional perturbations to each sample. We consider both high and low-frequency noise on each channel independently. For high-frequency noise, we first generate a noisy sequence $\mathbf{s}$, which has the same length as the signal channel, and each element of $\mathbf{s}$ is i.i.d. sampled from a uniform distribution $U[-1, 1]$. We then control the amplitude of $\mathbf{s}$ by the noisy degree $d \in \mathbb{R}$. Finally, we add the scaled noisy sequence $d \cdot \mathbf{s}$ to the channel. In the experiment, $d = 0.05$ for Sleep-EDF, $d = 0.002$ for HAR, $d = 0.001$ for PTB-XL and $d = 0.01$ for MGH. For low-frequency noise, we generate a short noisy sequence (the length is randomly sampled from a uniform distribution $U[100, \text{length of channel}]$) in the same way and then use *scipy.interpolate.interp1d* to interpolate the noisy sequence to be at the same length as the channel. The choice of high-frequency noise or low-frequency noise, or both are coin-tossed with equal probability. (ii) *Bandpass filtering* reduces signal noise. We use the order-1 Butterworth filter by *scipy.signal.butter* to preserve only the within-band frequency information. The high-pass and low-pass are $(1Hz, 30Hz)$ and $(10Hz, 49Hz)$ for Sleep-EDF, $(1Hz, 20Hz)$ and $(5Hz, 24.5Hz)$ for HAR, $(1Hz, 30Hz)$ and $(10Hz, 50Hz)$ for PTB-XL, $(1Hz, 30Hz)$ and $(10Hz, 50Hz)$ for MGH. Low-pass or high-pass or both are selected with equal probability. Also, the bandpass filtering is applied to each channel independently. The intuition is that the low-pass signals and high-pass signals might be both useful. (iii) *3D position rotation* is an augmentation technique used only for HAR datasets, which have x-y-z axis information from accelerometer and gyroscope sensors. We apply a 3D x-y-z coordinate system rotation by a rotation matrix to mimic different cellphone positions. All augmentation methods are applied in sequence (i) (ii) (iii). The STFT is performed after the data augmentation.

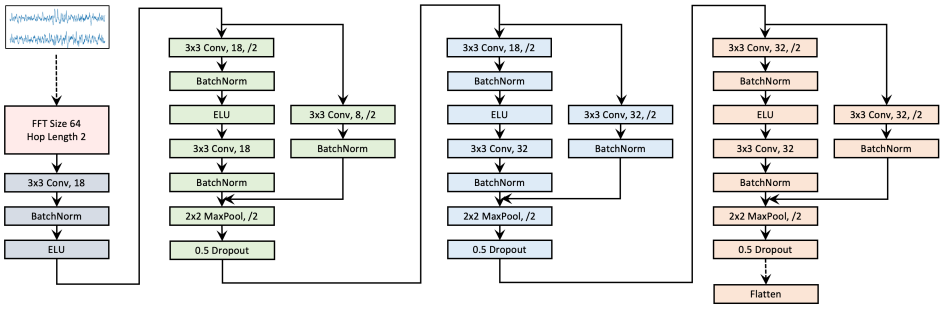

Figure 4: Backbone CNN Architecture (HAR). The architectures for other three datasets are similar.

C.4 Implementation

Since all deep learning baselines are based on CNN, we use the same backbone model, shown in Figure 4. The model is adopted from (Cheng et al., 2020). Based on the backbone model, we add a fully connected layer for the reference CNN model, add non-linear layers for self-supervised models (they also have their respective loss), and add corresponding deconvolutional layers for autoencoder models. The reference CNN models is end-to-end, and they are trained on the training set; other baselines learn a low-dimensional feature extractor from an unlabeled set, and then a logistic classifier is trained on the training set, on top of the feature extractor. Without loss of generality, we use 128 as batch size. For deep learning models, we use Adam optimizer with a learning rate 1×10^{-3} and weight decaying 5×10^{-4}. We use 2×10^{-3} as the learning rate for our `ATD`. Since the paper deals with unsupervised learning and uses standard logistic regression

(*sklearn.linear_model.LogisticRegression*) to evaluate, the hyperparameters of all models (except supervised model) are chosen based on the classification performance on the training data. For our ATD model, by default, we use $R = 32, \alpha = 1 \times 10^{-3}, \beta = 2, \gamma = b$ (which is the batch size), and all the reference configurations for each experiments are listed in code appendix. The choice of R depends on the trade-off between model fitness and time complexity. Specifically, a larger R means better fitness and more preserved information in the extracted representations, while the number of learnable parameters and time complexity also increases linearly with R. In our paper, we run simple CP decomposition on a small subset of the tensor and monitor the fitness curve. We find that the fitness does not improve much around $R = 32$ for all datasets (which means the real tensor might have a smaller rank and the residual part might be just noise). Thus, we choose $R = 32$ throughout the paper.

C.5 Ablation Studies on Data Augmentation

Effect of High-quality Data Augmentations. We study the effects of high-quality data augmentation methods. Let us assume *(a): jittering*, *(b): bandpass filtering*, *(c): 3D position rotation*. We test on different combinations of the augmentation methods. The experiments are conducted on two datasets: (i) for the MGH dataset, we use 50,000 unlabeled data, 5,000 training samples, and all test samples; (ii) for the HAR dataset, we use all data. We use 128 as the batch size for MGH Sleep, 64 for HAR, and $R = 32$ as the tensor rank.

Table 6: Ablation Studies on Data Augmentation (MGH Sleep)

Method	**Acc (%)**	**Method**	**Acc (%)**	**Method**	**Acc (%)**
(a)	72.53 ± 0.652	(b)	73.60 ± 0.270	(a)+(b)	74.15 ± 0.203

Table 7: Ablation Studies on Data Augmentation (HAR)

Method	**Acc (%)**	**Method**	**Acc (%)**	**Method**	**Acc (%)**
(a)	92.06 ± 0.621	(a)+(b)	92.70 ± 0.266	(a)+(b)+(c)	93.35 ± 0.357
(b)	91.99 ± 0.274	(a)+(c)	92.93 ± 0.590	/	/
(c)	92.29 ± 0.330	(b)+(c)	92.26 ± 0.479	/	/

Table 6 and Table 7 conclude that the augmentation methods influence the final classification results. However, for different datasets, the effects are different. We observe that for the MGH Sleep dataset, bandpass filtering works better than jittering. In HAR, combinations of augmentations work better than the individual augmentations. Overall, we find that with more diverse data augmentation methods, the final results are relatively better. The study of how to choose/design (or even automatically generate) better augmentation techniques will be our future work.

Impact of Low-quality Data Augmentations. We also study the impact of low-quality data augmentations. We use the SALS model (mini-batch tensor baseline without data augmentation) and our `ATD` as the reference models and conduct the following experiments on MGH and HAR:

- MGH-d: we change the d values for the jittering data augmentation, d means the ratio of the amplitude of the high/low frequency noise over the amplitude of the signal.
- MGH-(A): on MGH data, we set the degree of the *jittering* method to an unrealistic value $d = 1$ as the low-quality augmentation method, meaning that the magnitude of the noise is the same as the magnitude of the signals. We keep the *bandpass filtering* unchanged.
- MGH-(B): on MGH data, we randomly create a *bandpass filter* from $(1Hz, 10Hz)$ or $(30Hz, 50Hz)$ as the low-quality augmentation method, which drops the critical middle-band information. We keep the *jittering* unchanged.
- MGH-(C): on MGH data, we combine the above two low-quality augmentation methods.
- HAR-(A)(B)(C): follow the similar low-quality data augmentation method design on HAR. For (B), we use $(1Hz, 5Hz)$ and $(20Hz, 24.5Hz)$ as the low-quality bandpass.

Table 8: Performance of Low-quality Data Augmentation (on MGH and HAR)

MGH Accuracy	$d = 0.02$	$d = 0.05$	$d = 0.1$	$d = 0.5$	$d = 5$
	74.18 ± 0.326	74.10 ± 0.302	73.85 ± 0.530	73.39 ± 0.493	72.18 ± 0.676
MGH Accuracy	**SALS**	**ATD**	**(A)**	**(B)**	**(C)**
	71.93 ± 0.379	74.15 ± 0.431	72.73 ± 0.624	72.10 ± 0.719	70.75 ± 0.771
HAR Accuracy	**SALS**	**ATD**	**(A)**	**(B)**	**(C)**
	91.86 ± 0.295	93.35 ± 0.357	92.04 ± 0.308	92.48 ± 0.469	91.43 ± 0.835

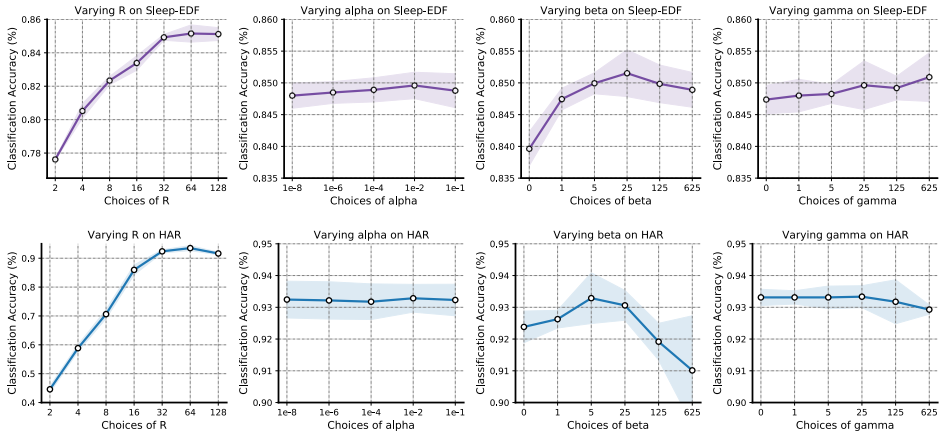

Figure 5: Ablation Studies on Hyperparameters

Table 9: Comparison with Tensor-based Methods with Different R (on HAR)

Model	$R = 8$	$R = 16$	$R = 32$	$R = 64$	$R = 128$
SALS	69.59 ± 0.526	83.92 ± 0.416	91.84 ± 0.295	91.89 ± 0.217	91.55 ± 0.388
GR-SALS	69.62 ± 0.458	84.20 ± 0.727	92.33 ± 0.282	92.28 ± 0.359	91.84 ± 0.534
$\texttt{ATD}_{ss-}$	70.27 ± 0.488	84.84 ± 0.557	92.41 ± 0.391	92.71 ± 0.243	92.32 ± 0.330
ATD	71.91 ± 0.253	85.61 ± 0.294	93.35 ± 0.357	93.43 ± 0.411	92.97 ± 0.273

The performances are shown in Table 8. We find that low-quality data augmentations will hurt the learned tensor bases and hinder the downstream classification. With d changing from 0.02 to 5, the *jittering* method becomes more unrealistic and the generated samples deviate from the real signal data distribution. Thus, we can find that the final classification performance also becomes worse gradually. If all data augmentation methods are of low-quality, the performance cannot surpass the base SALS model. We also find that the performance drop is not significant. The reason might be that even the augmentation methods are of low-quality, the Frobenius loss can still enforce the model to learn decent subspaces for better fitness and find generic low-rank features (opposed to the classification-oriented low-rank features), in this case, the features only follow fitness principle not the alignment principle, so the performance will be similar compared to SALS.

C.6 Ablation Studies on Hyperparameters

This section conducts ablation studies for decomposition rank R and other hyperparameters, α, β, γ. The experiments are conducted on Sleep-EDF with 50,000 random unlabeled data, 5,000 random training samples, and all test samples, and the HAR dataset.

The results are shown in Figure 5. First, we can conclude that with a larger decomposition rank R, the performance will be better generally. Though we observe that the performance worsens from $R = 64$ to $R = 128$, with limited training data, if the representation size (equals to R) becomes larger, the logistic regression model can overfit. We compare our model to other tensor-based methods with

Table 10: Comparison with Tensor-based Methods with Different R (on Sleep-EDF)

Model	$R = 8$	$R = 16$	$R = 32$	$R = 64$	$R = 128$
SALS	81.26 ± 0.345	82.59 ± 0.638	84.27 ± 0.481	84.55 ± 0.527	84.49 ± 0.317
GR-SALS	81.72 ± 0.664	82.74 ± 0.481	84.33 ± 0.356	84.87 ± 0.486	84.90 ± 0.781
$\texttt{ATD}_{ss-}$	81.27 ± 0.568	82.5 ± 0.674	84.19 ± 0.221	84.47 ± 0.258	84.44 ± 0.577
ATD	82.49 ± 0.464	83.31 ± 0.591	85.01 ± 0.224	85.30 ± 0.483	85.32 ± 0.305

Table 11: Result Significance and Running Time (%) for Sleep-EDF and HAR

	Sleep-EDF (5,000)			**HAR (1,473)**		
	Accuracy	# of Params.	Time per sweep	Accuracy	# of Params.	Time per sweep
Self-sup models:						
SimCLR-32	84.98 ± 0.358	210,384	260.299s	74.75 ± 0.723	53,286	8.459s
SimCLR-128	$\mathbf{85.19 \pm 0.358}$	222,768	265.809s	76.69 ± 0.697	65,670	8.532s
BYOL-32	84.29 ± 0.405	211,440	255.614s	73.71 ± 2.832	54,342	8.430s
BYOL-128	83.26 ± 0.337	239,280	257.266s	71.79 ± 1.866	82,182	8.478s
Auto-encoders:						
AE-32	74.78 ± 0.723	217,216	153.684s	63.13 ± 0.775	62,940	7.530s
AE-128	75.17 ± 0.897	241,888	156.813s	60.52 ± 1.604	87,612	7.662s
AE_{ss}-32	80.92 ± 0.345	217,216	301.773s	71.70 ± 2.135	62,940	7.765s
AE_{ss}-128	81.84 ± 0.259	241,888	307.546s	72.43 ± 1.370	87,612	7.804s
Tensor models:						
SALS	84.27 ± 0.481 (0.0041)	7,328	86.281s	91.86 ± 0.295 (2e-5)	2,688	7.535s
GR-SALS	84.33 ± 0.356 (0.0019)	7,328	109.916s	92.33 ± 0.282 (0.0003)	2,688	7.829s
$\texttt{ATD}_{ss-}$	84.19 ± 0.221 (9e-5)	7,328	147.568s	92.41 ± 0.391 (0.0011)	2,688	8.604s
ATD	85.01 ± 0.224	7,328	148.375s	$\mathbf{93.35 \pm 0.357}$	2,688	8.672s

result format: mean $\pm$ standard deviation (p-value)

$R = 8, 16, 32, 64, 128$ and the Table 9 and Table 10 shows that ATD outperforms the tensor baselines consistently with different R.

Also, we find that the choices of α do not affect the final performance a lot. Finally, we find that it is easy for users to select the β values from a large range in general. For example, selecting a $\beta \in [5, 125]$ would guarantee good results on Sleep-EDF while on HAR, the selection range is $[5, 25]$. The choice of β does affect the final performance, but it is not tricky to search for a β for good performance. We also find that the accuracy score first increases then decreases with an increasing value of β. The reason might be that a large β will negatively affect the fitness loss.

C.7 Statistical Testing and Running Time Comparison

In this section, we conduct T-test on the result in main text and calculate the p-values in the parenthesis of Table 11 and Table 12 (the experimental results are copied from Table 2). Commonly, a p-value smaller than 0.05 would be considered as significant. We can see that our model show significant performance gain over all baselines on all tasks.

We have also reported the running time per sweep/epoch in the tables. When recording the running time, we duplicated the environment mentioned in Section 4.1, stopped other programs and ran all the models one by one on GPUs. We record the first 8 sweeps/epochs of all models and drop the first 3 sweeps (since they might be unstable). The average running time of the last 5 epochs are reported in Table 11 and Table 12 while the accuracy results are from Table 2. Note that on HAR, PTB-XL and Sleep-EDF, all methods use 128 as the batch size and on MGH, all methods use 512 as the batch size. The tensor based methods all use $R = 32$ as the rank. We can conclude that the tensor based methods are generally more time-efficient than the deep learning methods with fewer parameters. Since our model ATD and the variant $\texttt{ATD}_{ss-}$ use the augmented tensors, they cost more compared to other tensor based methods (since the size of training tensors doubles), however, we also observe empirically that they can converge faster with around half number of the epochs.

C.8 Comparison with Supervised Tensor Learning

In this subsection, we compare our model with two supervised tensor learning baselines, UMLDA Lu et al. (2008) and supervised tensor learning (STL) Tao et al. (2005). **UMLDA** extracts uncorrelated discriminative features by sequential tensor-to-vector projections, and it includes two stages: in the

Table 12: Result Significance and Running Time (%) for PTB-XL and MGH

	PTB-XL (2,183)			MGH (5,000)		
	Accuracy	# of Params.	Time per sweep	Accuracy	# of Params.	Time per sweep
Self-sup models:						
SimCLR-32	69.25 ± 0.355	200,960	18.714s	67.34 ± 0.970	212,624	1449.368s
SimCLR-128	68.19 ± 0.793	237,920	19.037s	66.98 ± 1.331	246,608	1457.283s
BYOL-32	65.08 ± 1.535	202,016	18.410s	68.83 ± 1.168	214,736	1451.468s
BYOL-128	65.49 ± 0.612	254,432	18.680s	68.55 ± 1.339	279,632	1461.181s
Auto-encoders:						
AE-32	59.01 ± 0.896	224,528	11.229s	68.58 ± 0.427	220,088	851.118s
AE-128	58.29 ± 0.412	298,352	11.396s	67.05 ± 1.375	257,048	815.858s
AE_{ss}-32	68.47 ± 0.231	224,528	18.263s	71.46 ± 0.386	220,088	1486.244s
AE_{ss}-128	68.88 ± 0.604	298,352	18.465s	70.19 ± 0.617	257,048	1504.545s
Tensor models:						
SALS	69.15 ± 0.483 (0.0023)	7,296	8.988s	71.93 ± 0.379 (5e-6)	9,984	782.763s
GR-SALS	69.02 ± 0.477 (0.0012)	7,296	9.747s	72.35 ± 0.228 (8e-6)	9,984	970.292s
$\texttt{ATD}_{ss-}$	69.38 ± 0.612 (0.0129)	7,296	12.560s	72.78 ± 0.522 (0.0005)	9,984	1327.188s
ATD	**70.26 ± 0.523**	7,296	12.599s	**74.15 ± 0.431**	9,984	1360.569s

result format: mean ± standard deviation (p-value)

Table 13: Comparison with Supervised Tensor Learning

Model	Sleep-EDF (5,000)	HAR (1,473)	PTB-XL (2,183)	MGH (5,000)
UMLDA (supervised pretrain + supervised LR)	81.06 ± 0.093	85.73 ± 1.169	65.55 ± 0.267	62.04 ± 0.722
STL (end-to-end supervised)	77.86 ± 0.816	80.52 ± 0.189	61.83 ± 0.712	41.44 ± 0.597
ATD (unsupervised pretrain + supervised LR)	**85.01 ± 0.224**	**93.35 ± 0.357**	**70.26 ± 0.523**	**74.15 ± 0.431**

first stage (supervised pre-training stage), it uses label information to maximize the Fisher's discrimination criterion (FDC) as the objective and extract uncorrelated features (i.e., representations); in the second stage (supervised learning stage), it uses another supervised model to map the representations to the labels. To make a fair comparison, we use 32 as the uncorrelated feature dimension (thus, it has the same number of learnable parameters as our model) and also use logistic regression (LR) for the second stage. **STL** is an end-to-end supervised tensor learning baseline, and it is originally proposed for binary classification with a rank-one parameterized tensor (i.e., outer product of multiple vectors). In the comparison, we extend STL for mult-class classification by including more rank-one parameterized tensors (one for each class). We use cross entropy loss to optimize the revised STL model. Both baselines are implemented with PyTorch and use the training and test set only. We have carefully turned the baseline models to achieve higher accuracy.

Result Analysis. The results are shown in Table 13. We can conclude that our methods outperform these two supervised tensor learning baselines significantly. The performance gap between UMLDA and our model can be explained by (i) UMLDA uses FDC criterion to design the loss function and also forces the learned feature dimensions to be uncorrelated, which might discard some essential class-dependent information; (ii) our model utilizes a large set of unlabeled data. STL gives poor performance because it is essentially a multilinear method with much fewer parameters than UMLDA and our ATD, which hurts the expressive.

D Batch-based Optimization Algorithm

We have shown our optimization algorithm for handling smaller tensors in the main text. Here, we present the batch-based algorithm for handling larger tensors that are optimized batch-by-batch in Algorithm 2.

Algorithm 2: Mini-batch Alternating Least Squares

Input: Data tensor $\mathcal{T} \in \mathbb{R}^{N \times I \times J \times K}$; initialized $\{\mathbf{A}^1, \mathbf{B}^1, \mathbf{C}^1\}$; batch size b; learning rate η; other hyperparameters α, β, γ; initial counter $l = 1$;
repeat
  shuffle the data tensor $\mathcal{T}$; /* start a new sweep */
  for *a tensor batch* $\mathcal{T}^l \in \mathbb{R}^{b \times I \times J \times K}$ *and its augmentation* $\tilde{\mathcal{T}}^l$ **do**
    use $\mathbf{A}^l, \mathbf{B}^l, \mathbf{C}^l$ to initialize $\mathbf{X}$ based on $\mathcal{T}^l$;
    use $\mathbf{A}^l, \mathbf{B}^l, \mathbf{C}^l$ to initialize $\tilde{\mathbf{X}}$ based on $\tilde{\mathcal{T}}^l$;
    Use $\mathbf{A}^l, \mathbf{B}^l, \mathbf{C}^l, \tilde{\mathbf{X}}$ to update $\mathbf{X}$ by our iterative rules (one iteration) in Eqn. (15);
    Use $\mathbf{A}^l, \mathbf{B}^l, \mathbf{C}^l, \mathbf{X}$ to update $\tilde{\mathbf{X}}$ by our iterative rules (one iteration) in Eqn. (15);
    Use $\mathbf{B}^l, \mathbf{C}^l, \mathbf{X}, \tilde{\mathbf{X}}$ to obtain $\mathbf{A}^{l+1}$ by solving least square problem;
    Use $\mathbf{A}^{l+1}, \mathbf{C}^l, \mathbf{X}, \tilde{\mathbf{X}}$ to obtain $\mathbf{B}^{l+1}$ by solving least square problem;
    Use $\mathbf{A}^{l+1}, \mathbf{B}^{l+1}, \mathbf{X}, \tilde{\mathbf{X}}$ to obtain $\mathbf{C}^{l+1}$ by solving least square problem;
    $l = l + 1$ /* increment the counter */;
  end
until *max sweep exceeds or change of loss* $< 0.1\%$ *within 3 consecutive sweeps*;
Output: the learned bases $\{\mathbf{A}^L, \mathbf{B}^L, \mathbf{C}^L\}$.

E Derivation of Two-sided Bound

We recall the definition of $\mathcal{L}_{ss}$ and $\mathcal{L}_{ss}^{\Theta}$ from Section 3.1.

$$\begin{aligned}\mathcal{L}_{ss} =\ & \mathcal{L}_{pos} + \lambda \mathcal{L}_{neg} \\ =\ & \mathbb{E}\left[\frac{\lambda}{1-r_p}\text{sim}\left(\mathbf{f}\left(\mathcal{X}_p\right), \mathbf{f}\left(\mathcal{Y}_q\right)\right)\right] - \mathbb{E}\left[\left(\frac{\lambda r_p}{1-r_p} + 1\right)\text{sim}\left(\mathbf{f}\left(\mathcal{X}_p\right), \mathbf{f}\left(\mathcal{Y}_q\right)\right) \mid p = q\right], \\ \mathcal{L}_{ss}^{\Theta}(\gamma) =\ & (\gamma+1)\mathbb{E}\left[\text{sim}\left(\mathbf{f}\left(\mathcal{X}_p\right), \mathbf{f}\left(\mathcal{Y}_q\right)\right)\right] - \mathbb{E}\left[\text{sim}\left(\mathbf{f}\left(\mathcal{X}_p\right), \mathbf{f}\left(\mathcal{Y}_q\right)\right) \mid p = q\right],\end{aligned}$$

and we want to prove that

$$C_1\mathcal{L}_{ss}^{\Theta}\left(\frac{\lambda-1}{C_1}\right) \leq \mathcal{L}_{ss} \leq C_2\mathcal{L}_{ss}^{\Theta}\left(\frac{\lambda-1}{C_2}\right),\ C_1 = 1 + \max_p \frac{\lambda r_p}{1-r_p}, C_2 = 1 + \min_p \frac{\lambda r_p}{1-r_p},$$

where r_p is the label rate of class-p.

Proof. We start by arranging $\mathcal{L}_{ss}$,

$$
\begin{aligned}
\mathcal{L}_{ss} &= \mathbb{E}\left[\frac{\lambda}{1-r_p}\text{sim}\left(\mathbf{f}\left(\mathcal{X}_p\right),\mathbf{f}\left(\mathcal{Y}_q\right)\right)\right] - \mathbb{E}\left[\left(\frac{\lambda r_p}{1-r_p}+1\right)\text{sim}\left(\mathbf{f}\left(\mathcal{X}_p\right),\mathbf{f}\left(\mathcal{Y}_q\right)\right) \mid p=q\right] \\
&= \mathbb{E}\left[\left(\frac{\lambda r_p}{1-r_p}+\lambda\right)\text{sim}\left(\mathbf{f}\left(\mathcal{X}_p\right),\mathbf{f}\left(\mathcal{Y}_q\right)\right)\right] - \mathbb{E}\left[\left(\frac{\lambda r_p}{1-r_p}+1\right)\text{sim}\left(\mathbf{f}\left(\mathcal{X}_p\right),\mathbf{f}\left(\mathcal{Y}_q\right)\right) \mid p=q\right] \\
&= \mathbb{E}\left[\left(\frac{\lambda r_p}{1-r_p}+1\right)\text{sim}\left(\mathbf{f}\left(\mathcal{X}_p\right),\mathbf{f}\left(\mathcal{Y}_q\right)\right)\right] + \mathbb{E}\left[(\lambda-1)\,\text{sim}\left(\mathbf{f}\left(\mathcal{X}_p\right),\mathbf{f}\left(\mathcal{Y}_q\right)\right)\right] \\
&\quad - \mathbb{E}\left[\left(\frac{\lambda r_p}{1-r_p}+1\right)\text{sim}\left(\mathbf{f}\left(\mathcal{X}_p\right),\mathbf{f}\left(\mathcal{Y}_q\right)\right) \mid p=q\right] \qquad (16) \\
&= \mathbb{E}_p\mathbb{E}_{q,\mathcal{X}_p,\mathcal{Y}_q}\left[\left(\frac{\lambda r_p}{1-r_p}+1\right)\text{sim}\left(\mathbf{f}\left(\mathcal{X}_p\right),\mathbf{f}\left(\mathcal{Y}_q\right)\right)\right] + \mathbb{E}\left[(\lambda-1)\,\text{sim}\left(\mathbf{f}\left(\mathcal{X}_p\right),\mathbf{f}\left(\mathcal{Y}_q\right)\right)\right] \\
&\quad - \mathbb{E}_p\mathbb{E}_{q,\mathcal{X}_p,\mathcal{Y}_q}\left[\left(\frac{\lambda r_p}{1-r_p}+1\right)\text{sim}\left(\mathbf{f}\left(\mathcal{X}_p\right),\mathbf{f}\left(\mathcal{Y}_q\right)\right) \mid p=q\right] \qquad (17) \\
&= \mathbb{E}_p\left[\left(\frac{\lambda r_p}{1-r_p}+1\right)\mathbb{E}_{q,\mathcal{X}_p,\mathcal{Y}_q}\left[\text{sim}\left(\mathbf{f}\left(\mathcal{X}_p\right),\mathbf{f}\left(\mathcal{Y}_q\right)\right)\right]\right] + \mathbb{E}\left[(\lambda-1)\,\text{sim}\left(\mathbf{f}\left(\mathcal{X}_p\right),\mathbf{f}\left(\mathcal{Y}_q\right)\right)\right] \\
&\quad - \mathbb{E}_p\left[\left(\frac{\lambda r_p}{1-r_p}+1\right)\mathbb{E}_{q,\mathcal{X}_p,\mathcal{Y}_q}\left[\text{sim}\left(\mathbf{f}\left(\mathcal{X}_p\right),\mathbf{f}\left(\mathcal{Y}_q\right)\right) \mid p=q\right]\right] \\
&= \mathbb{E}_p\left[\left(\frac{\lambda r_p}{1-r_p}+1\right)\left(\mathbb{E}_{q,\mathcal{X}_p,\mathcal{Y}_q}\left[\text{sim}\left(\mathbf{f}\left(\mathcal{X}_p\right),\mathbf{f}\left(\mathcal{Y}_q\right)\right)\right] - \mathbb{E}_{q,\mathcal{X}_p,\mathcal{Y}_q}\left[\text{sim}\left(\mathbf{f}\left(\mathcal{X}_p\right),\mathbf{f}\left(\mathcal{Y}_q\right)\right) \mid p=q\right]\right)\right] \\
&\quad + \mathbb{E}\left[(\lambda-1)\,\text{sim}\left(\mathbf{f}\left(\mathcal{X}_p\right),\mathbf{f}\left(\mathcal{Y}_q\right)\right)\right] \qquad (18) \\
&\leq \mathbb{E}_p\left[C_2\left(\mathbb{E}_{q,\mathcal{X}_p,\mathcal{Y}_q}\left[\text{sim}\left(\mathbf{f}\left(\mathcal{X}_p\right),\mathbf{f}\left(\mathcal{Y}_q\right)\right)\right] - \mathbb{E}_{q,\mathcal{X}_p,\mathcal{Y}_q}\left[\text{sim}\left(\mathbf{f}\left(\mathcal{X}_p\right),\mathbf{f}\left(\mathcal{Y}_q\right)\right) \mid p=q\right]\right)\right] \\
&\quad + \mathbb{E}\left[(\lambda-1)\,\text{sim}\left(\mathbf{f}\left(\mathcal{X}_p\right),\mathbf{f}\left(\mathcal{Y}_q\right)\right)\right] \qquad (19) \\
&= (C_2+\lambda-1)\mathbb{E}\left[\text{sim}\left(\mathbf{f}\left(\mathcal{X}_p\right),\mathbf{f}\left(\mathcal{Y}_q\right)\right)\right] - C_2\mathbb{E}\left[\text{sim}\left(\mathbf{f}\left(\mathcal{X}_p\right),\mathbf{f}\left(\mathcal{Y}_q\right)\right) \mid p=q\right] \\
&= C_2\mathcal{L}_{ss}^{\Theta}\left(\frac{\lambda-1}{C_2}\right).
\end{aligned}
$$

From Eqn. (16) to Eqn. (17), we use the fact that "$\mathbb{E}[\cdot]$ means the expectation is taken over four interdependent random variables, i.e., $p, q, \mathcal{X}_p, \mathcal{Y}_q$", which is mentioned in Section 3.1. From Eqn. (18) to Eqn. (19), we use the fact that given p, the similarity of random pairs is smaller than the similarity of positive pairs $\mathbb{E}_{q,\mathcal{X}_p,\mathcal{Y}_q}\left[\text{sim}\left(\mathbf{f}\left(\mathcal{X}_p\right),\mathbf{f}\left(\mathcal{Y}_q\right)\right)\right] \leq \mathbb{E}_{q,\mathcal{X}_p,\mathcal{Y}_q}\left[\text{sim}\left(\mathbf{f}\left(\mathcal{X}_p\right),\mathbf{f}\left(\mathcal{Y}_q\right)\right) \mid p=q\right]$. The upper bound is derived by replacing $\frac{\lambda r_p}{1-r_p}+1,\ \forall p$ with $C_2 = 1+\min_p \frac{\lambda r_p}{1-r_p}$. Similarly, we can also derive the other side (lower bound) by using $C_1 = 1+\max_p \frac{\lambda r_p}{1-r_p}$, which eventually gives $C_1\mathcal{L}_{ss}^{\Theta}\left(\frac{\lambda-1}{C_1}\right)$. □