Appendix: Fast Algorithms for Incremental and Decremental Semi-supervised Discriminant Analysis

Wenrao Pang and Gang Wu

Appendix A. Notations

Table 1: Some notations used in the paper.

Notations	Descriptions
d, n	Data dimension and number of samples
l, u	Number of labeled or unlabeled samples
β_1, β_2	Regularization parameters
c	The number of classes
X^L, X^U	Data matrix composed of labeled or unlabeled samples
S, L	The similarity matrix and the corresponding Laplacian matrix
S_B	The between-class scatter matrix
$S_T, \overline{S}_T$	The total scatter matrix and the modified total scatter matrix
r_t, r_b	The number of desired eigenpairs of $\overline{S}_T$ and S_B
P, G	The matrix containing leading eigenvectors of $\overline{S}_T$ and S_B
Π, Δ	The diagonal matrix containing leading eigenvalues of $\overline{S}_T$ and S_B
X^{inc}	The matrix composed of added samples
X_L^{inc}	The matrix composed of added labeled samples
X_U^{inc}	The matrix composed of added unlabeled samples
X^{dec}	The matrix composed of decremental samples
$\overline{S}_T^{new}, S_B^{new}$	The updated modified total scatter matrix and the modified between-class scatter matrix
Φ	Sufficient spanning set represented by the set of orthogonal basis

Appendix B. Proof of Theorem 1

Theorem 1. *Let $X = QR$ be the economized QR decomposition of X, where $Q \in \mathbb{R}^{d\times n}$ is an orthonormal matrix, and $R \in \mathbb{R}^{n\times n}$ is upper triangular. If $(\lambda, \mathbf{y})$ is an eigenpair of the n-by-n matrix $RR^T + \beta_1 RLR^T + \beta_2 I$, i.e.,*

$$\big(RR^T + \beta_1 RLR^T + \beta_2 I\big)\mathbf{y} = \lambda\mathbf{y}, \tag{0.1}$$

then $(\lambda, Q\mathbf{y})$ is an eigenpair of the d-by-d matrix $\overline{S}_T$.

Proof. Let $\overline{S}_T\mathbf{v} = \lambda\mathbf{v}$, and let $X = QR$ be the economized QR decomposition of X, so we have

$$\big(QRR^TQ^T + \beta_1 QRLR^TQ^T + \beta_2 I\big)\mathbf{v} = \lambda\mathbf{v},$$

which can be rewritten as

$$\big(RR^TQ^T + \beta_1 RLR^TQ^T + \beta_2 Q^T\big)\mathbf{v} = \lambda Q^T\mathbf{v}. \tag{0.2}$$

For *any* vector $\mathbf{v} \in \mathbb{R}^d$, we have that $\mathbf{v} = QQ^T\mathbf{v} + Q_\perp Q_\perp^T\mathbf{v}$, where $Q_\perp$ is an orthonormal basis for the orthogonal complement of the subspace span$\{Q\}$, such that $\big[Q,\ Q_\perp\big]$ is a unitary matrix. It follows from (0.2) that

$$\big(RR^TQ^T + \beta_1 RLR^TQ^T + \beta_2 Q^T\big)(QQ^T\mathbf{v} + Q_\perp Q_\perp^T\mathbf{v}) = \lambda Q^T(QQ^T\mathbf{v} + Q_\perp Q_\perp^T\mathbf{v}). \tag{0.3}$$

Notice that $Q^TQ_\perp Q_\perp^T\mathbf{v} = 0$, then (0.3) can be formulated as

$$\big(RR^T + \beta_1 RLR^T + \beta_2 I\big)Q^T\mathbf{v} = \lambda Q^T\mathbf{v}. \tag{0.4}$$

Denote by $\tilde{S}_T = RR^T + \beta_1 RLR^T + \beta_2 I$ and by $\mathbf{y} = Q^T\mathbf{v}$. Thus, if $(\lambda, \mathbf{y})$ is an eigenpair of the n-by-n matrix $\tilde{S}_T$, then we have from (0.2)–(0.4) that $(\lambda, Q\mathbf{y})$ is an eigenpair of the d-by-d matrix $\overline{S}_T$. □

Appendix C. Analysis of equation (3.9)

Now we analyze Equation (3.9) in detail. For the first term, we have

$$\begin{aligned}
&\Phi^T(P_1\Pi_1P_1^T + \beta_1 X\cdot\Delta D_1\cdot X^T)\Phi \\
&= \begin{pmatrix} P_1^TP_1 \\ \Delta\Phi^TP_1 \end{pmatrix}\Pi_1\begin{pmatrix} P_1^TP_1 & P_1^T\Delta\Phi \end{pmatrix} + \beta_1(\Phi^TX)\Delta D_1(X^T\Phi) \\
&= \begin{pmatrix} \Pi_1 & \mathcal{O} \\ \mathcal{O} & \mathcal{O} \end{pmatrix} + \beta_1(\Phi^TX)\Delta D_1(X^T\Phi).
\end{aligned} \tag{0.5}$$

For the second term of Equation (3.9), we see that

$$\begin{aligned}
&\Phi^T(P_2\Pi_2P_2^T + \beta_1 X^{inc}\Delta D_2(X^{inc})^T)\Phi \\
&= (\Phi^TP_2)\Pi_2(P_2^T\Phi) + \beta_1(\Phi^TX^{inc})\Delta D_2((X^{inc})^T\Phi).
\end{aligned} \tag{0.6}$$

The third term $\Phi^T(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^T\Phi$ is a rank-one matrix, we only need to calculate the matrix-vector product $\Phi^T(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)$ for it. For the last term, we have

$$\begin{aligned}&\Phi^T\left(XS_{12}(X^{inc})^T+X^{inc}S_{12}^TX^T\right)\Phi\\&=(\Phi^TX)S_{12}((X^{inc})^T\Phi)+(\Phi^TX^{inc})S_{12}^T(X^T\Phi).\end{aligned}\tag{0.7}$$

In summary, the main overhead is to compute the matrix-vector products Φ^TX, Φ^TP_2, Φ^TX^{inc}, as well as $\Phi^T(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)$ for approximating the $(r_{t_1}+r_{t_2}+1)$-by-$(r_{t_1}+r_{t_2}+1)$ matrix $\Phi^T\overline{S}_T^{new}\Phi$, with $r_{t_1}+r_{t_2}+1\ll d$.

Appendix D. Computational eigenmodels of Ξ

For the sake of completeness, we also present the algorithm for computing eigenmodels Ξ of the between scatter matrix S_B, for more details on this algorithm, refer to [18, Sec. 2.2]. It is used in our proposed algorithms (see Algorithm 2 and Algorithm 3 in the main paper).

Algorithm 1 An algorithm for eigenmodels of the between-class scatter S_B [18]

Input: Labeled data matrix $X^L=[\mathbf{x}_1^L,\mathbf{x}_2^L,\ldots,\mathbf{x}_l^L]\in\mathbb{R}^{d\times l}$;

Output: The number l, n_k, vectors $\boldsymbol{\mu}^L$, $\boldsymbol{\alpha}_k$ and the matrices G and Δ.

1. Let n_k be the number of samples in the k-th class;

2. Compute the mean vector $\boldsymbol{\mu}^L$ of the labeled samples, and the mean vectors of the labeled samples in each class: $\boldsymbol{m}_1,\ldots,\boldsymbol{m}_C$;

3. Compute the centralized matrix $H=\left[\sqrt{n_1}(\boldsymbol{m}_1-\boldsymbol{\mu}_L),\ldots,\sqrt{n_C}(\boldsymbol{m}_C-\boldsymbol{\mu}_L)\right]$;

4. Compute the eigen-decomposition of the matrix $H^TH=V\Sigma\Sigma^TV^T$, and let $\Sigma_{r_b}=diag(\sigma_1,\ldots,\sigma_{r_b})$ be the matrix whose diagonals are composed of the r_b dominant eigenvalues, and V_{r_b} be the matrix composed of the r_b dominant eigenvectors, where r_t is number of the required eigenpairs;

5. Let $G=HV_{r_b}\Sigma_{r_b}^{-1}\in\mathbb{R}^{d\times r_b}$ and $\Delta=\Sigma_{r_b}\Sigma_{r_b}^T\in\mathbb{R}^{r_b\times r_b}$,

6. Let $\boldsymbol{\alpha}_k=G^T(\boldsymbol{m}_k-\boldsymbol{\mu}^L)$, where $k=1,2,\ldots,c$, and c is the number of classes;

7. Let the model $\Xi=\left\{\boldsymbol{\mu}^L,l,G,\Delta,n_k,\boldsymbol{\alpha}_k|k=1,2,\ldots,c\right\}$.

Appendix E. A comparison of the computational cost and storage requirements of the four algorithms: ISSDA, IncSDA, DecSDA and DecUSDA.

We briefly compare the time complexities and storage requirements of ISSDA [5], our incremental algorithm IncSDA (Algorithm 2), and the two decremental algorithms DecSDA (Algorithm 3) and DecUSDA (Algorithm 4).

In the initial training process, the main computational cost of ISSDA is in $\mathcal{O}(d^3)$ flops, and the storage requirements are in $\mathcal{O}(n^2)$, where d is the data dimension, n is the number of training samples. As a comparison, the main computational cost of IncSDA is in about $\mathcal{O}\big(d(n^2+c^2+r_b^2)+n^3\big)$ flops, where $\mathcal{O}\big(dn^2+n^3\big)$ flops for Algorithm 1 (of the main paper), and $\mathcal{O}\big(d(c^2+r_b^2)\big)$ flops for the algorithm in **Appendix D.** For storage requirements, we only need to pay $\mathcal{O}\big(d(n+r_b+c)+n^2\big)$ for IncSDA. Thus, if $d \gg n$ (which happens in the small-sample-size problem), IncSDA is much cheaper than ISSDA in initial training.

In the incremental learning process, we have to pay $\mathcal{O}(d(n^2+r_b^2))$ flops for computation and $\mathcal{O}\big(d(n+r_b+r_t)+n^2\big)$ for storage in ISSDA. As a comparison, it costs us $\mathcal{O}\big(d(n^2+r_b^2+r_t^2)\big)$ flops for computation and $\mathcal{O}\big(d(n+r_b+r_t)+n^2\big)$ for storage in IncSDA. Thus, the incremental learning process of ISSDA is a little cheaper than that of IncSDA. This is due to the fact that we have to update both the total scatter matrix and the between-class scatter matrix in the new method. For the two decremental algorithms DecSDA and DecUSDA, the computational cost and storage requirements are about the same as those of ISSDA. Please see Table 2 for a summary of the computational cost and storage requirements of the four algorithms ISSDA, IncSDA, DecSDA and DecUSDA.

Appendix F. Detail description of the datesets and the algorithms for comparison

- NIR-VIS-2.0 [1] [17] face data set consists of 5093 visible and 12,487 NIR

[1] http://www.cbsr.ia.ac.cn/english/NIR-VIS-2.0-Results.html

Table 2: *Appendix E: Computational cost and storage requirements of the four algorithms.*

Algorithm	**Computational cost**
Initial training:	
ISSDA	$\mathcal{O}(d^3)$
IncSDA	$\mathcal{O}\big(d(n^2+c^2+r_b^2)+n^3\big)$
Incremental learning:	
ISSDA	$\mathcal{O}(d(n^2+r_b^2))$
IncSDA	$\mathcal{O}(d(n^2+r_b^2+r_t^2))$
Initial training:	
DecSDA	$\mathcal{O}\big(d(n^2+c^2+r_b^2)+n^3\big)$
DecUSDA	$\mathcal{O}\big(d(n^2+c^2+r_b^2)+n^3\big)$
Decremental learning:	
DecSDA	$\mathcal{O}\big(d(n^2+r_b^2+r_t^2)\big)$
DecUSDA	$\mathcal{O}\big(d(n^2+r_b^2+r_t^2)\big)$
Algorithm	**Storage requirements**
Initial training:	
ISSDA	$\mathcal{O}(n^2)$
IncSDA	$\mathcal{O}\big(d(n+r_b+c)+n^2\big)$
Incremental learning:	
ISSDA	$\mathcal{O}\big(d(n+r_b+r_t)+n^2\big)$
IncSDA	$\mathcal{O}\big(d(n+r_b+r_t)+n^2\big)$
Initial training:	
DecSDA	$\mathcal{O}\big(d(n+r_b+c)+n^2\big)$
DecUSDA	$\mathcal{O}\big(d(n+r_b+c)+n^2\big)$
Decremental learning:	
DecSDA	$\mathcal{O}\big(d(n+r_b+r_t)+n^2\big)$
DecUSDA	$\mathcal{O}\big(d(n+r_b+r_t)+n^2\big)$

frontal images, involving a total of 725 subjects. Specifically, VIS contains 719 valid subjects and NIR contains 721 valid subjects for experiments. The dimension of the raw images is 128×128.

• The Extended Yale Face Database B [2] contains 16,128 images of 28 human subjects under 9 poses and 64 illumination conditions [9]. The data format of this database is the same as the Yale Face Database B. The extended database as opposed to the original Yale Face Database B with 10 subjects was first re-

[2] `http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html`

ported by Lee *et al.* [19]. The "cropped" Extended YaleB contains 38 subjects with 64 images to per people. We also apply the face detector MATLAB function *CascadeObjectDetector.m* in the MATLAB Vision package to the "original" Extended YaleB includes 28 classes, generating images with 168×192 pixels, and we call the resulting data set the "self-cropped" Extended YaleB.

• The YouTube Faces DataBase[3] [33] is a large-scale video classification database which contains 1,595 persons/classes. In the experiments, we use 48 to 90 samples in each class. Consequently, the number of subset of YouTube Faces dataset is 124,819 and each facial image is vectorized into an self-cropped image of size 160×160, obtained from running the MATLAB function *CascadeObjectDetector.m* in the MATLAB Vision package.

The following algorithms are used for comparison on incremental or decremental learning problems, whose details are described as follows:

• Semi-supervised discriminant analysis (**SDA**) [1] aims to find a projection which respects the discriminant structure inferred from the labeled data points, as well as the intrinsic geometrical structure inferred from both labeled and unlabeled data points.

• Incremental semi-supervised discriminant analysis (**ISSDA**) method [5] aims at reducing the computational complexity by updating only S_B incrementally. However, it only considers unlabeled samples when forming the scatter matrix in the initial training, and does not update the total scatter matrix.

• Incremental linear discriminant analysis (**ILDA**) [17] is an supervised incremental learning method. It makes use of the sufficient spanning sets method to divide the incremental model solution into three steps for approximation, and leads an online solution which closely agrees with the batch solution in accuracy while significantly reducing the computational complexity. The algorithm yields an efficient solution to incremental LDA even when the number of classes as well as the set size is large.

• Incremental principal component analysis (**IPCA**) [37] is an unsupervised

[3] `https://www.cs.tau.ac.il/~wolf/ytfaces/`

incremental learning method, which is based on the incremental SVD method. This algorithm can update the main subspace without simply recalculating the eigendecomposition from scratch. In all the experiments, we reserve 95% of the energy in the IPCA method.

• Semi-supervised discriminant embedding (**SDE**) [7] is a semi-supervised extension of local discriminant embedding (LDE). SDE can overcome the difficulty of the small-sample-size (SSS) problem in some degree, and emphasize the discrimination property by enlarging the distances between data that belong to different classes.

• Incremental exponential semi-supervised discriminant embedding (**IESDE**) [22] is a fast incremental method for semi-supervised exponential embedding (**ESDE**) [7]. In this method, the added samples can be labeled or unlabeled.

• Spectral regression discriminant analysis in semi-supervised version (**SemiSRDA**) [2] takes advantage of both labeled and unlabeled samples, to learn the discriminant function as smooth as possible on the data manifold. It is based on the spectral regression technique for the semi-supervised learning problem.

Appendix G. Supplementary experiments

G1. Supplementary experiments of Example 5.1 and Example 5.2

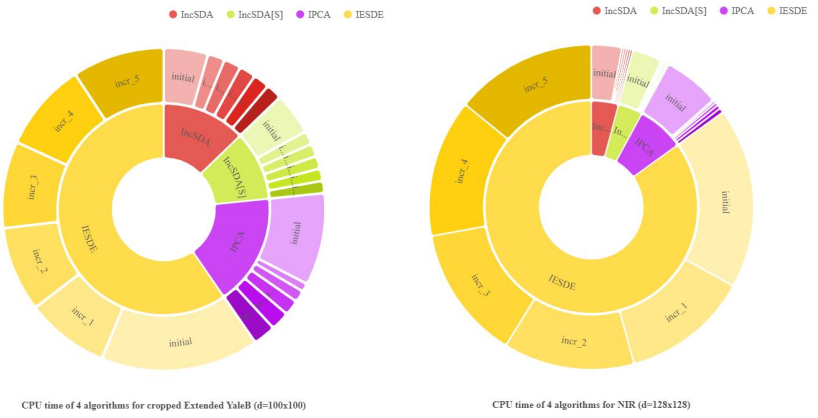
In Section 5.1, we also consider the "balanced scenario" during the incremental procedure. The data set is the NIR-VIS-2.0 data set with dimension $d = 128 \times 128$. So as to maintain class balance in the initial and the incremental training process, the initial batch consists of one image per subject and each incremental batch consists of one new image per subject. Thus, there are k images in both initial and each incremental batch, where k is the number of subjects. Similar operations are for the test sets. The remaining images are used as unlabeled ones for training.

From the results in Table 3, we observe that the recognition accuracy of our proposed IncSDA method is the highest among the five algorithms, both for initial training and the incremental training processes. Moreover, for the high-

dimensional data set NIR-VIS-2.0, the CPU time used in IncSDA is comparable to the IPCA method, and it runs much faster than ISSDA and IESDE. That is, the proposed algorithm is very competitive to incremental learning problems arising from high-dimensional data set.

Table 3: **Example 5.1**: *CPU time in seconds (Recognition rate, Macro avg F1-score) and the total CPU time in seconds of the algorithms IncSDA (Algorithm 2), ILDA, IPCA, ISSDA and IESDE on the NIR-VIS-2.0 database, with d=*128×128*. "Initial" denotes the results obtained from the initial training and "Incr x" stands for those from the x-th incremental experiment, and "Total CPU" denotes total CPU time for initial training and the 3 incremental experiments.*

VIS	Initial	Incr 1	Incr 2	Incr 3	Total CPU
ILDA	0.3694(92.79%, 0.9078)	0.5556(92.79%, 0.9078)	0.6284(92.63%, 0.9054)	0.6231(92.60%, 0.9049)	2.1765
IPCA	18.7453(45.27%, 0.4359)	1.5163(46.28%, 0.4499)	2.0205(45.78%, 0.4435)	2.6127(45.30%, 0.4390)	24.8948
IncSDA	5.7130(97.35%, 0.9655)	4.3534(97.76%, 0.9705)	4.5640(97.88%, 0.9720)	4.7515(97.97%, 0.9733)	19.3818
ISSDA	561.3118(94.84%, 0.9332)	1.9303(94.84%, 0.9332)	2.5845(95.20%, 0.9376)	2.5723(95.39%, 0.9398)	568.3989
IESDE	20.6210(84.94%, 0.8542)	16.5916(89.21%, 0.8759)	21.8303(89.76%, 0.8787)	29.1961(90.12%, 0.8864)	88.2390
NIR	Initial	Inc 1	Inc 2	Inc 3	Total CPU
ILDA	0.3450(65.01%, 0.5946)	0.5031(65.01%, 0.5946)	0.5518(65.61%, 0.6006)	0.5493(65.99%, 0.6044)	1.9491
IPCA	51.7690(0.14%, 0.0004)	3.0315(0.21%, 0.0011)	4.1491(0.17%, 0.0009)	5.3282(0.19%, 0.0010)	64.2778
IncSDA	23.8805(81.26%, 0.7663)	5.3055(82.19%, 0.7778)	5.5549(82.52%, 0.7819)	5.6449(82.83%, 0.7860)	40.3858
ISSDA	558.4535(74.70%, 0.6937)	1.5993(74.70%, 0.6937)	2.1342(76.03%, 0.7086)	2.1300(76.91%, 0.7188)	564.3170
IESDE	107.5659(55.82%, 0.4312)	84.5610(50.29%, 0.4247)	84.9408(47.77%, 0.4056)	104.1870(47.98%, 0.4248)	381.2547



a) b)

Fig. 1. *Example 5.2: Total CPU time (in seconds) of the four algorithms on incremental semi-supervised problem with unlabeled samples: a) the cropped Extended YaleB database with* $d = 100 \times 100$ *and b) the NIR database with* $d = 128 \times 128$.

G2. Statistical tests for the comparison of the examined algorithms

In order to perform statistical tests for the comparison of the examined algorithms, in each experiment, we make use of the STAC platform [26] [4] to verify the results obtained from the learning algorithms. According to the STAC assistant decision tree [26], we choose Friedman Aligned Ranks (with the significance level of 0.05) to access the examined algorithms. As was pointed out in the STAC platform, the lower the result of an algorithm on a problem, the better the performance of such algorithm. Hence, we use

$$Error = 1 - Recognition\ rate \tag{0.8}$$

as the import data in STAC.

Table 4 lists the Friedman Aligned Ranks computed from the STAC web platform, using Error ($= 1 - Accuracy\ rate$) in Table 2 from paper as the import data. It is seen from the statistical tests that IncSDA ranks the highest among the compared algorithms.

Table 5 list the Friedman Aligned Ranks computed from the STAC web platform, using Error ($= 1 - Accuracy\ rate$) in Table 3 as the import data, respectively. It is seen from the statistical tests that IncSDA ranks the highest among the compared algorithms.

Table 6 lists the Friedman Aligned Ranks computed from the STAC web platform, using Error ($= 1 - Accuracy\ rate$) in Table 3 from paper as the import data. We observe from the statistical tests that IncSDA ranks the highest among the compared algorithms.

Table 7 lists the Friedman Aligned Ranks computed from the STAC web platform, using Error ($= 1 - Accuracy\ rate$) in Table 4 from paper as the import data. It is seen from the statistical tests that IncSDA and IncSDA[S] rank about the third or the fourth among the compared algorithms.

[4] `https://tec.citius.usc.es/stac/index.html`. We thank one of the reviewers for providing us with the link.

Table 4: **Example 5.1**: *Statistical tests for comparisons of the examined algorithms: Friedman Aligned Ranks from the STAC web platform*

Dataset	Algorithm	Rank	Comparison	Statistic	Adjusted p-value	Result
			ISSDA vs IncSDA	4.41	0.00	Rejected
Cropped	**IncSDA**	3.5	ISSDA vs IESDE	2.94	0.02	Rejected
Extended YaleB	IESDE	9.5	IncSDA vs ILDA	2.94	0.02	Rejected
(d=32 × 32)	ILDA	15.5	ISSDA vs ILDA	1.47	0.85	Accepted
	ISSDA	21.5	IESDE vs IncSDA	1.47	0.85	Accepted
			IESDE vs ILDA	1.47	0.85	Accepted
	IncSDA	5.7	IncSDA vs IPCA	4.30	0.00	Rejected
			ILDA vs IPCA	3.38	0.01	Rejected
	ILDA	10.3	IncSDA vs IESDE	2.52	0.12	Accepted
Cropped			ISSDA vs IPCA	2.36	0.18	Accepted
Extended YaleB	ISSDA	15.5	ISSDA vs IncSDA	1.93	0.53	Accepted
(d=64 × 64)			IESDE vs IPCA	1.77	0.77	Accepted
	IESDE	18.5	IESDE vs ILDA	1.61	1.00	Accepted
			ISSDA vs ILDA	1.02	1.00	Accepted
	IPCA	27.5	IncSDA vs ILDA	0.92	1.00	Accepted
			ISSDA vs IESDE	0.59	1.00	Accepted
	IncSDA	6.0	IncSDA vs IPCA	4.23	0.00	Rejected
			IESDE vs IPCA	3.20	0.01	Rejected
	IESDE	11.3	ILDA vs IPCA	2.71	0.07	Accepted
Cropped			ISSDA vs IncSDA	2.56	0.11	Accepted
Extended YaleB	ILDA	13.8	ISSDA vs IPCA	1.67	0.94	Accepted
(d=100 × 100)			ISSDA vs IESDE	1.52	1.00	Accepted
	ISSDA	19.0	IncSDA vs ILDA	1.52	1.00	Accepted
			ISSDA vs ILDA	1.03	1.00	Accepted
	IPCA	27.5	IncSDA vs IESDE	1.03	1.00	Accepted
			IESDE vs ILDA	0.49	1.00	Accepted

Table 5: **Example 5.1**: *Statistical tests for comparisons of the examined algorithms: Friedman Aligned Ranks from the STAC web platform, using Error* $(= 1 - Accuracy\ rate)$ *in Table 3 as the import data.*

Dataset	Algorithm	Rank	Comparison	Statistic	Adjusted p-value	Result
VIS	**IncSDA**	3.0	IncSDA vs IPCA	4.30	0.00	Rejected
			ISSDA vs IPCA	3.22	0.01	Rejected
	ISSDA	8.0	IncSDA vs IESDE	3.22	0.01	Rejected
			ISSDA vs IESDE	2.15	0.32	Accepted
	ILDA	13.0	IncSDA vs ILDA	2.15	0.32	Accepted
			ILDA vs IPCA	2.15	0.32	Accepted
	IESDE	18.0	ISSDA vs IncSDA	1.07	1.00	Accepted
			IESDE vs ILDA	1.07	1.00	Accepted
	IPCA	23.0	IESDE vs IPCA	1.07	1.00	Accepted
			ISSDA vs ILDA	1.07	1.00	Accepted
NIR	**IncSDA**	2.5	IncSDA vs IPCA	3.82	0.00	Rejected
			ISSDA vs IPCA	2.87	0.04	Rejected
	ISSDA	6.5	IncSDA vs IESDE	2.87	0.04	Rejected
			ISSDA vs IESDE	1.91	0.56	Accepted
	ILDA	10.5	IncSDA vs ILDA	1.91	0.56	Accepted
			ILDA vs IPCA	1.91	0.56	Accepted
	IESDE	14.5	ISSDA vs IncSDA	0.96	1.00	Accepted
			ISSDA vs ILDA	0.96	1.00	Accepted
	IPCA	18.5	IESDE vs ILDA	0.96	1.00	Accepted
			IESDE vs IPCA	0.96	1.00	Accepted

Table 6: **Example 5.2**: *Statistical tests for comparisons of the examined algorithms: Friedman Aligned Ranks from the STAC web platform.*

Dataset	Algorithm	Rank	Comparison	Statistic	Adjusted p-value	Result
Cropped Extended YaleB (d=100 × 100)			IPCA vs IncSDA	3.67	0.00	Rejected
	IncSDA	6.5	IPCA vs IESDE	2.86	0.03	Rejected
	IESDE	9.8	IncSDA[S] vs IPCA	2.29	0.13	Accepted
	IncSDA[S]	12.2	IncSDA[S] vs IncSDA	1.39	0.99	Accepted
	IPCA	21.5	IncSDA vs IESDE	0.82	1.00	Accepted
			IncSDA[S] vs IESDE	0.57	1.00	Accepted
NIR (d=128 × 128)			IPCA vs IncSDA	1.70	0.54	Accepted
	IncSDA	3.3	IncSDA vs IESDE	1.59	0.68	Accepted
	IncSDA[S]	6.3	IncSDA[S] vs IncSDA	1.02	1.00	Accepted
	IESDE	8.0	IncSDA[S] vs IPCA	0.68	1.00	Accepted
	IPCA	8.3	IncSDA[S] vs IESDE	0.57	1.00	Accepted
			IPCA vs IESDE	0.11	1.00	Accepted

Table 7: **Example 5.3**: *Statistical tests for comparisons of the examined algorithms: Friedman Aligned Ranks from the STAC web platform.*

Dataset	Algorithm	Rank	Comparison	Statistic	Adjusted p-value	Result
	IESDE	3.5	IESDE vs ILDA	4.72	0.00	Rejected
			IncSDA[S] vs IESDE	3.54	0.00	Rejected
	SemiSRDA	10.0	SemiSRDA vs ILDA	3.44	0.01	Rejected
self-cropped			IncSDA vs ILDA	2.46	0.14	Accepted
YouTubeFace	**IncSDA**	15.0	IncSDA[S] vs SemiSRDA	2.26	0.24	Accepted
(d=160 × 160)			IncSDA vs IESDE	2.26	0.24	Accepted
	IncSDA[S]	21.5	IncSDA[S] vs IncSDA	1.28	1.00	Accepted
			SemiSRDA vs IESDE	1.28	1.00	Accepted
	ILDA	27.5	IncSDA[S] vs ILDA	1.18	1.00	Accepted
			SemiSRDA vs IncSDA	0.98	1.00	Accepted
	SemiSRDA	3.5	SemiSRDA vs IESDE	4.72	0.00	Rejected
			SemiSRDA vs IncSDA	3.54	0.00	Rejected
	ILDA	12.2	IESDE vs ILDA	3.02	0.03	Rejected
self-cropped			IncSDA[S] vs IESDE	2.89	0.04	Rejected
Extended YaleB	**IncSDA[S]**	12.8	IncSDA[S] vs SemiSRDA	1.84	0.66	Accepted
(d=168 × 192)			IncSDA vs ILDA	1.84	0.66	Accepted
	IncSDA	21.5	SemiSRDA vs ILDA	1.71	0.88	Accepted
			IncSDA[S] vs IncSDA	1.71	0.88	Accepted
	IESDE	27.5	IncSDA vs IESDE	1.18	1.00	Accepted
			IncSDA[S] vs ILDA	0.13	1.00	Accepted

Table 8 lists the Friedman Aligned Ranks computed from the STAC web platform, using Error ($= 1 - Accuracy\ rate$) in Table 5 and Table 6 from paper as the import data, respectively. We see from the statistical tests that DecUSDA and DecSDA rank the second and the third among the compared algorithms.

Table 8: **Example 5.4**: *Statistical tests for comparisons of the examined algorithms: Friedman Aligned Ranks from the STAC web platform.*

Dataset	Algorithm	Rank	Comparison	Statistic	Adjusted p-value	Result
Cropped Extended YaleB (incorrect labels)	SemiSRDA	3.8	SemiSRDA vs SDA	4.46	0.00	Rejected
			SemiSRDA vs SDE	3.48	0.01	Rejected
	DecUSDA	10.4	SDA vs DecUSDA	3.16	0.02	Rejected
			DecSDA vs SemiSRDA	2.25	0.25	Accepted
	DecSDA	15.3	DecSDA vs SDA	2.21	0.27	Accepted
			DecUSDA vs SDE	2.18	0.29	Accepted
	SDE	21.5	SemiSRDA vs DecUSDA	1.30	1.00	Accepted
			DecSDA vs SDE	1.23	1.00	Accepted
	SDA	26.5	SDA vs SDE	0.98	1.00	Accepted
			DecSDA vs DecUSDA	0.95	1.00	Accepted
Cropped Extended YaleB (noise samples)	**DecUSDA**	4.0	DecUSDA vs DecSDA	0.58	0.56	Accepted
	DecSDA	5.0				