ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
"ΠΛΗΡΟΦΟΡΙΚΗ"

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Εκφράζοντας τις πολυγραμμικές μεθόδους PCA και LDA ως προβλήματα ελαχίστων τετραγώνων

Χρήστος Μ. Χατζής

ΕΠΙΒΛΕΠΩΝ: **Ιωάννης Παναγάκης,** Αναπληρωτής Καθηγητής, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

ΑΘΗΝΑ

ΑΠΡΙΛΙΟΣ 2022

NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND THELECOMMUNICATIONS

POSTGRADUATE STUDY PROGRAMME
"COMPUTER SCIENCE"

MASTER THESIS

A Least Squares Formulation for Multilinear PCA and LDA

Christos M. Chatzis

SUPERVISOR: **Ioannis Panagakis,** Associate Professor, National and Kapodistrian University of Athens

ATHENS

APRIL 2022

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Εκφράζοντας τις πολυγραμμικές μεθόδους PCA και LDA ως προβλήματα ελαχίστων τετραγώνων

Χρήστος Μ. Χατζής

Α.Μ.: CS1200004

ΑΠΡΙΛΙΟΣ 2022

MASTER THESIS

A Least Squares formulation for Multilinear PCA and LDA

Christos M. Chatzis

ID.: CS1200004

APRIL 2022

ΠΕΡΙΛΗΨΗ

Αν και η πρώτη ερευνητική δραστηριότητα σχετικά με την Ανάλυση Συνιστωσών (Component Analysis - CA) εμφανίστηκε αρκετές δεκαετίες πριν, ο τομέας αυτός είναι ακόμη αρκετά ενεργός. Δοσμένου ενός συνόλου δεδομένων, μία μέθοδος CA υπολογίζει μια απεικόνιση (mapping) των αρχικών δεδομένων, στην οποία τα χαρακτηριστικά κάθε δείγματος θα εξυπηρετούν καλύτερα τα διαθέσιμα εργαλεία και τον εκάστοτε σκοπό. Συνήθως, η προκύπτουσα προβολή έχει λιγότερα χαρακτηριστικά από το σύνολο εισόδου και συνεπώς η προσέγγιση αυτή ειναι γνωστή και ως Μείωση Διαστάσεων (Dimensionality Reduction). Παρόλο που αυτοί οι μέθοδοι ήταν αρχικά σχεδιασμένοι για διανυσματικά δεδομένα, η ανάγκη για ανάλυση πολυδιάστατων δεδομένων αποτέλεσε όχημα για την επέκταση τους σε τανυστές. Σε αυτήν την διπλωματική εργασία, θα εστιάσουμε σε δύο τέτοιες επεκτάσεις: την Πολυγραμμική Ανάλυση Κύριων Συνιστωσών (Multilinear Principal Component Analysis – MPCA) και την Ανάλυση Διάκρισης με Αναπαράσταση Τανυστή (Discriminant Analysis with Tensor Representation – DATER) και θα παρουσιάσουμε πώς διατυπώνονται ως προβλήματα εύρεσης ιδιοτιμών και ιδιοδιανυσμάτων. Μια τέτοια διατύπωση, ωστόσο, εμπεριέχει τα εξής προβλήματα: (1) δεν απαγορεύει την επίλυση προβλημάτων εύρεσης ιδιοτιμών και ιδιοδιανυσμάτων σε πίνακες κακής κατάστασης (ill-conditioned matrices), πράγμα που ισχύει αρκετά συχνά σε δεδομένα τανυστών [1] και (2) οι εμπλεκόμενοι πίνακες έχουν μεγάλες διαστάσεις και η επίλυση τέτοιων προβλημάτων απαιτεί αρκετό χρόνο. Για το σκοπό αυτό, προτείνουμε έναν τρόπο διατύπωσης των MPCA και DATER ως προβλήματα Παλινδρόμησης Τανυστών, έτσι ώστε να μπορούν να εφαρμοστούν περισσότερο αριθμητικά ευσταθείς και υπολογιστικά απλούστερες προσεγγίσεις (π.χ. Gradient Descent). Κατόπιν, εξετάζουμε την ποιότητα της πρότασης μας σε πραγματικά δεδομένα με πείραματα Αφαίρεσης Θορύβου (Image denoising) και Αναγνώρισης Προσώπου (Face recognition).

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Μηχανική Μάθηση

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Πολυγραμμική Ανάλυση Κύριων Συνιστωσών, Ανάλυσης Διάκρισης με Αναπαράσταση Τανυστή, Αποσυνθέσεις Τανυστών, Μέθοδοι Τανυστών, Παλινδρόμηση Τανυστών

ABSTRACT

Although the first works relevant to Component Analysis (CA) date many decades ago, it still remains a very active research area. Given a dataset, CA methods aim to find a mapping of it, the features of which are ideal for the available tools or the assigned task. Typically, the produced mapping has fewer features than the original data, therefore this approach is also known as Dimensionality Reduction. While these methods were designed to work on vectors, the need to analyze multidimensional datasets with an abundance of features, fueled their extension to tensors. In this thesis, we will investigate two such extensions, Multilinear Principal Component Analysis (MPCA) and Discriminant Analysis with Tensor Representation (DATER) and present how they are formulated as generalized eigenproblems. Such formulation, however, conceals several drawbacks: (1) it may require solving eigenproblems on ill-conditioned matrices, which is more than often the case when it comes to tensor data [1], and (2) the matrices involved are commonly highly dimensional and solving for their eigenvalues requires significant computation time. To this end, we will propose a Least Squares (LS) Tensor Regression formulation for MPCA and DATER, which makes applicable more numerically stable and computationally simpler approaches (e.g., Gradient Descent) and evaluate it in practice with an Image denoising and Face recognition task.

SUBJECT AREA: Machine Learning

KEYWORDS: Multilinear Principal Component Analysis, Discriminant Analysis with Tensor Representation, Tensor Decompositions, Tensor Methods, Tensor Regression

“We must know. We will know.”

- *David Hilbert*

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor Professor Ioannis Panagakis. His input was always on point and his directions were crucial to this work. I am also extremely grateful to my family for all their love and support throughout this journey.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Component analysis

With the ability to capture and store datasets with abundant features nowadays, Component Analysis (CA) methods are becoming more and more widely used. In their essence, these methods produce a mapping of the input data to a more "meaningful" subspace. Of course, the exact interpretation of "meaningful" depends on the method used and the task at hand.

Principal Component Analysis (PCA) [2, 3], for example, creates uncorrelated features that capture as much variance from the input points as possible. Mathematically, PCA aims to find an orthogonal transformation such that this criterion is satisfied. From a geometric point of view, PCA can be seen as a transformation of the axis, so that each direction captures the most possible variation, and each axis is perpendicular to each other. The new axes are referred to as Principal Axes, while the projection of the features on these axes as Principal Components. A visual intuition is presented in Figure 1. We note that PCA does not discard feature per se, but instead it creates new features from linear combinations of the original.

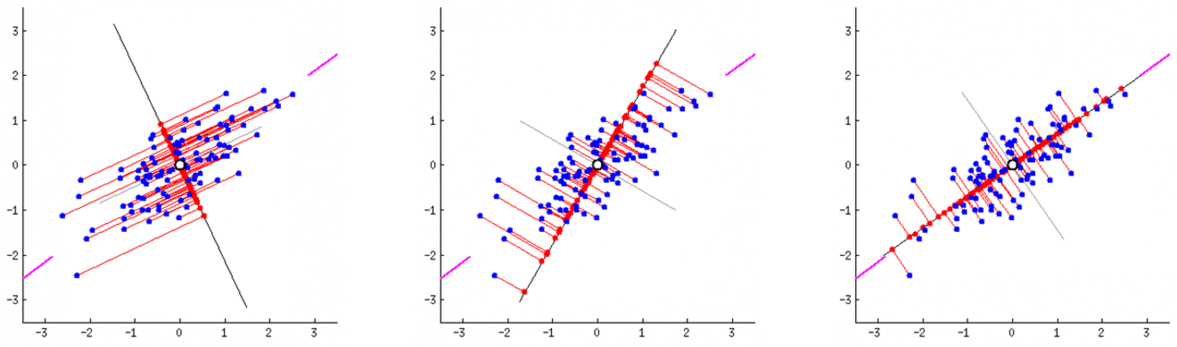

Figure 1: Finding the optimal 1D Principal Axis on sample 2D Data. Blue and red dots denote samples and projection, respectively. PCA computes the Principal Axis (black line) so that it captures as much as variance as possible, therefore the optimal solution is the rightmost. The gray line denotes the second Principal Axis [4].

On the other hand, Linear Discriminant Analysis (LDA) [5] takes a supervised approach. More specifically, this method projects the input data to the subspace that maximizes the distance between (the means of) the classes, while keeping the distance between samples of a class (and the respective class mean) at minimum. For the binary case, this objective is famously known as "Fisher's Discrimination Criterion" and computing the respective linear transformation is typically a trace quotient problem, as we will investigate in the following.

Independent Component Analysis (ICA) [3, 6] is an unsupervised approach that aims to represent the data in statistically independent features. While this approach may seem similar to PCA, we should note that ICA has a generative approach, in the sense that it estimates a linear combination of random variables that produces the input data. ICA is famously associated with the "Cocktail Party Problem", according to which one is tasked with distinguishing between speakers in a party, where supposedly multiple people talk simultaneously. Furthermore, the required statistical independency of the produced features is a stricter constraint that the uncorrelatedness of PCA. In other words, ICA prohibits any kind of relationship between produced features while PCA just linear ones.

Canonical Correlation Analysis (CCA) [7] is useful when the task at hand reduces in in modeling the interactions between two sets of data. Specifically, CCA receives two input data sets and projects them in the subspace, that maximizes the correlation between them. In practice, CCA attempts to produce features that represent both datasets as best as possible. These features, in a similar fashion to PCA, are referred to as Canonical Variables, while the bases of the constructed subspace as Canonical Axes.

It is not uncommon that data may hold a more complex structure and linear methods, such as the aforementioned, have poor performance. Consequently, kernel extensions have been proposed including KPCA [8], KLDA [9], KICA [10] and KCCA [11]. Moreover, non-linear embedding approaches are also frequently used, such as Laplacian Eigenmaps (LE) [12], Local Linear Embedding (LLE) [13], and ISOMAP [14]. These methods share a common rationale: if two samples are “close” (as in similar) to each other, their projections should be “close” too.

Evidently, incorporating such methods yields many benefits. To begin with, they can be used as a feature extraction technique, in order to avoid the computational burden of highly correlated features or produce more meaningful ones when the input is sparse (i.e., lots of zeros) or when discrimination is important (e.g., LDA). What’s more, the hardware prerequisites for storing and working on the downsized dataset are decreased, as less memory and computations are required. All these, along with the fact that these methods typically involve only a small number of parameters make them all the more appealing.

1.2 Extending to tensors

Lately, there has been a lot of focus on developing effective component analysis methods on data that evolves or is produced “naturally” over multiple axis. Indicative examples of such data are videos, with the axes being width, height and time and functional Magnetic Resonance Images (fMRI) with the addition of depth. Tensors, or multi-dimensional arrays, are becoming steadily an indispensable tool when dealing with such data, because not only they possess intuitive structure (spontaneous extension of matrices), but also many operations used by modern Machine Learning approaches extend naturally to them [15]. For example, if we model each frame of a video as 2D array, with each entry denoting pixel intensity, we end up with a 3D tensor. These dimensions/axes are known as tensor modes and their quantity denotes the order of the tensor.

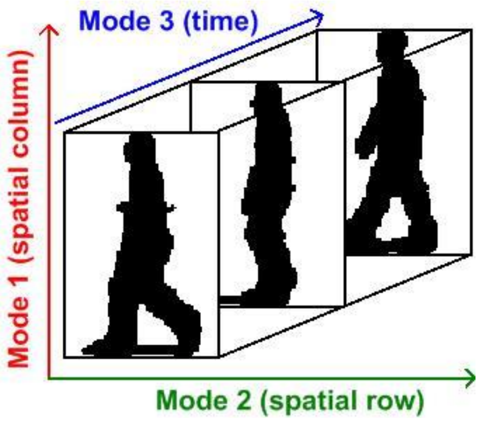

Figure 2: Representing frames of a sequence as a 3rd order tensor. Notice the three modes of such dataset [1].

One could easily notice that the tremendous increase in the number of features per tensor sample, so feature selection or dimensionality reduction is naturally advised. Moreover, the Small Sample Size problem (SSS)[1], which is usually the case when it comes to tensor data, contributes to ill-conditioning [1]. We could use one of the approaches mentioned in Section 1.1, after a vectorization operation. Albeit valid, this is approach is questionable. By doing so, we naively ignore the negative effects of SSS, amplify the consequences of the curse of dimensionality and additionally disregard information on how each part of the sample changes over all of the modalities. Tensor Component Analysis methods, on the other hand, such as those presented in the following, leverage this information, and produce more interpretable and numerically stable results.

Many approaches have been proposed on extending PCA for tensor data. Multilinear PCA (MPCA) [16] performs the PCA procedure mode-wise. Hence, the resulting projection consists of uncorrelated features in each mode. Uncorrelated MPCA (UMPCA) [17] projects each tensor sample to a vector while keeping the produced features (components of produced vectors) uncorrelated. Similarly, Tensor Rank-One Decomposition (TROD) [18] also produces a vector for each sample, but it does so by minimizing the reconstruction[2] error. Depending on the type of data, Non-negative MPCA (NMPCA) [19] may be also of interest. This method performs the same task as MPCA, while also enforcing non-negativity constraints. When outliers hinder the quality of the projection, Robust MPCA (RMPCA) [20] or the l_1-norm [21] have been proved as valuable countermeasures.

Many multilinear extensions for LDA have been suggested as well. Discriminant Analysis with Tensor Representation (DATER) [22] can be seen as applying LDA to each mode of the data. General Tensor Discriminant Analysis (GTDA) [23] has the same goal as DATER. However, the formulation of the GTDA is based on difference rather than a ratio. Uncorrelated Multilinear Discriminant Analysis (UMLDA) [24] and Tensor Rank-One Discriminant Analysis (TR1DA) [25] can be seen as the counterparts of UMPCA and TROD for LDA. Both methods project each tensor sample of the input to a vector with discriminative features, but UMLDA additionally employs uncorrelatedness constraints. The authors of [24] also propose R-UMLDA, which incorporates regularization, thus enhancing the method’s robustness.

Tensor-compatible ICA and CCA approaches have also been presented. Multilinear Mode-wise ICA (MMICA) [26] aims to uncover the mode-wise sources that produce the input data. 2D-CCA [27] and 3D-CCA [28] are the 2D and 3D extensions of the classical linear method. Contrary to the later proposed MCCA [29], however, they do not ensure uncorrelatedness of the projected features.

The Graph Embedding Framework [30] has also contributed to the development of TCA methods. According to the authors, input data shall be modeled in two weighted graphs: the “intrinsic” graph, which captures the similarity between data samples and the “penalty” graph, which describes the appropriate constraints. Any TCA method could be formulated under this framework as a generalized eigenvalue problem involving the Laplacian and adjacency matrices of these graphs. In addition to making the similarities and differences of TCA methods distinct, this work became the foundation for novel methods such as [31], [32] and [33].

[1] The SSS scenario is when the data contains more features than samples.

[2] Distance between original data and data after we apply the projection along with its inverse. We will thoroughly explore this notion throughout this work.

Before the end of this subsection, we should also mention the "Linear Subspace Learning" framework [1]. The authors, apart from providing an extensive survey, present a thorough categorization and in-detail technical review of the relevant methods.

1.3 Notation and preliminaries

From this point forward, scalars are denoted as lowercase letters (e.g., a, b). Vectors are denoted as lowercase bold letters (e.g., $\boldsymbol{a}, \boldsymbol{b}$), matrices as uppercase letters (e.g., A, B) and tensors as uppercase bold letters (e.g., $\boldsymbol{A}, \boldsymbol{B}$). The transpose and the inverse of a matrix A are denoted as A^T and A^{-1} respectively. Operators $tr(A)$ and $\|\boldsymbol{B}\|_F$ are used to denote the trace of matrix A and Frobenius norm of tensor $\boldsymbol{B}$.

The number of indexes required to access a single entry in a tensor defines its order. For example, vectors are of order 1, matrices are of order 2 and 3D tensors are of order 3. According to this, for tensor $\boldsymbol{A}$ of order 3, $\boldsymbol{A}_{i_1 i_2 i_3}$ denotes the element at position (i_1, i_2, i_3). For higher order tensors (at least 3 modalities) we can obtain its mode-n fibers by fixing all indices except the n-st. Slices can be obtained similarly by fixing the index of an additional mode. A visual illustration of fibers and slices for a 3D tensor is shown on Figures 2 and 3.

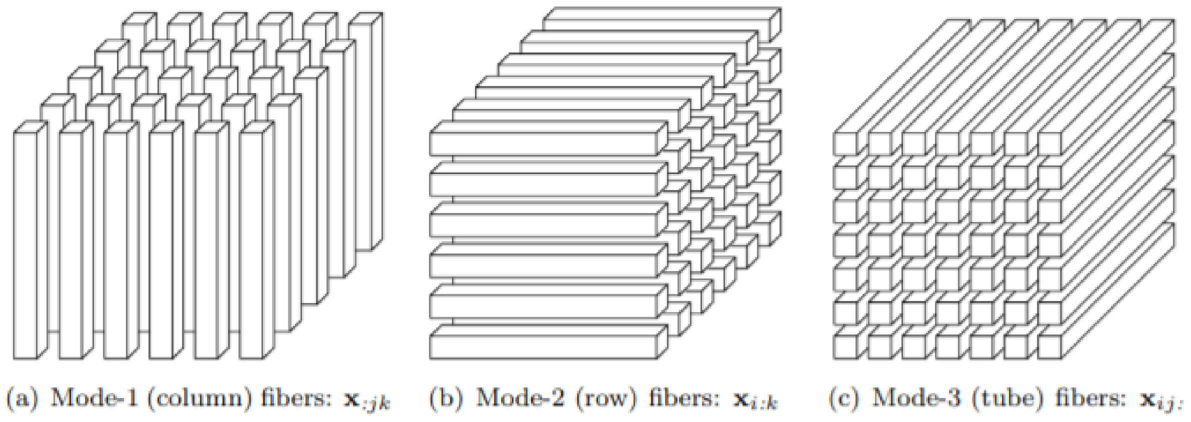
(a) Mode-1 (column) fibers: $\mathbf{x}_{:jk}$
(b) Mode-2 (row) fibers: $\mathbf{x}_{i:k}$
(c) Mode-3 (tube) fibers: $\mathbf{x}_{ij:}$

Figure 3: Fibers of 3D tensor [34].

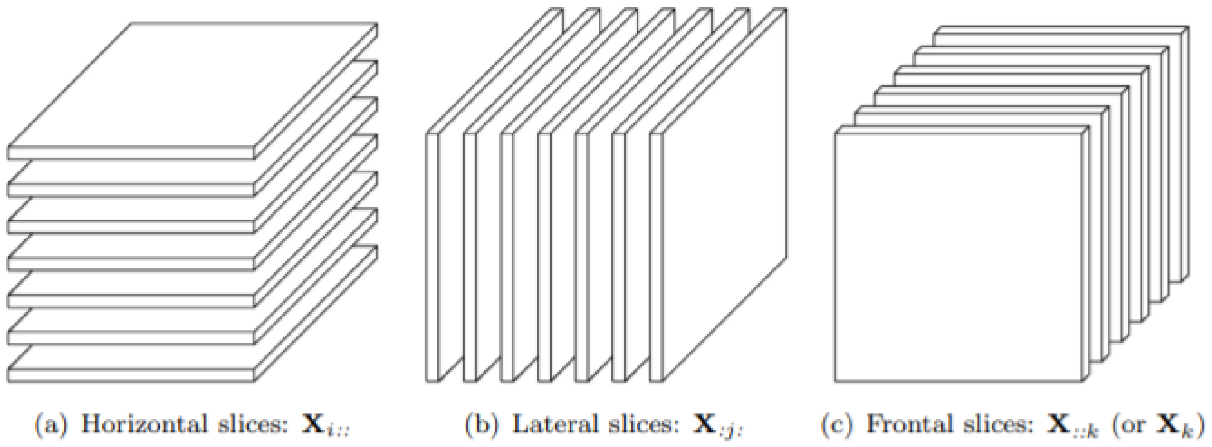
(a) Horizontal slices: $\mathbf{X}_{i::}$
(b) Lateral slices: $\mathbf{X}_{:j:}$
(c) Frontal slices: $\mathbf{X}_{::k}$ (or $\mathbf{X}_k$)

Figure 4: Slices of a 3D tensor [34].

It is often we require tensors in a matrix form. An organized way to do so is by tensor unfolding. Specifically, for a tensor $\boldsymbol{A} \in R^{p_1 \times p_2 \cdots \times p_n}$ we can define the mode-i unfolding, where the n-mode fibers are used in order form a matrix $\boldsymbol{A}_{(i)} \in R^{p_i \times p_1 \cdots p_{i-1} p_{i+1} \cdots p_n}$. More specifically, the element in position $(i_1, i_2, \dots, i_n)$ is mapped to the matrix's (i_i, j), where $j = 1 + \sum_{k=1, k \neq n}^{n} (i_k - 1) \times \prod_{m=k+1, m \neq i}^{n} p_m$. In the following, we will follow the definition of tensor unfolding given by [15]. Similarly, tensor vectorization for $\boldsymbol{A} \in R^{p_1 \times p_2 \cdots \times p_n}$ ($vec(\boldsymbol{A})$), results in a vector of size $R^{p_1 p_2 \cdots p_n}$.

We can now define the n-mode product between a matrix $M \in R^{q \times p_m}$ and a tensor $\boldsymbol{A}$:

$$\boldsymbol{A} \times_n M = M\boldsymbol{A}_{(n)} \in R^{p_n \times p_1 \cdots p_{n-1} q p_{n+1} \cdots p_m} \tag{1.1}$$

We note two important properties of the n-mode product that will be of use:

$$\boldsymbol{A} \times_n M \times_n N = \boldsymbol{A} \times_n NM \tag{1.2}$$

$$\boldsymbol{A} \times_n N \times_m M = \boldsymbol{A} \times_m M \times_n N \tag{1.3}$$

The inner product between tensors $\boldsymbol{A} \in R^{p_1 \times p_2 \cdots \times p_N}$ and $\boldsymbol{B} \in R^{p_1 \times p_2 \cdots \times p_n}$ is defined as:

$$< \boldsymbol{A}, \boldsymbol{B} >= \sum_{i_1=1}^{p_1} \sum_{i_2=1}^{p_2} \dots \sum_{i_n=1}^{p_n} \boldsymbol{A}_{i_1 i_2 \cdots i_n} \boldsymbol{B}_{i_1 i_2 \cdots i_n} \tag{1.4}$$

Likewise, the Generalized tensor inner product between tensors $\boldsymbol{A} \in R^{p_x \times p_1 \times p_2 \cdots \times p_n}$ and $\boldsymbol{B} \in R^{p_1 \times p_2 \cdots \times p_n \times p_Y}$ is:

$$< \boldsymbol{A}, \boldsymbol{B} >_n= \sum_{i_1=1}^{p_1} \sum_{i_2=1}^{p_2} \dots \sum_{i_n=1}^{p_n} \boldsymbol{A}_{: i_1 i_2 \cdots i_N} \boldsymbol{B}_{i_1 i_2 \cdots i_n :} \tag{1.5}$$

Notice that $< \boldsymbol{A}, \boldsymbol{B} >_n \in R^{p_x \times p_Y}$. In Table 1, we sum up the notation for various tensor related products that will be used throughout.

Table 1: Notation used for various tensor related products.

Symbol	Operation
$\circ$	Outer product
$\otimes$	Kronecker product
$\odot$	Khatri-Rao product
$<\cdot,\cdot>$	Inner product
$<\cdot,\cdot>_N$	Generalized inner product
$\cdot\times_n\cdot$	n-mode product

A tensor of n-th order that can be produced by the outer product of n vectors is referred to as a rank-1 tensor. In general, the rank of a n-th order tensor $\boldsymbol{X}$ is the minimum number of rank-1 tensors (also n-th order) that sum up to $\boldsymbol{X}$.

Finally, as a measure of variation for Tensors, we kindly remind readers of the total (1.6) and total i-mode scatter (1.7) matrix for a Tensor $\boldsymbol{X}$ with m samples:

$$\Psi_{\boldsymbol{A}} = S_{T_X} = \sum_{j=1}^{m} \left\| \boldsymbol{X}_j - \overline{\boldsymbol{X}} \right\|_F^2 \tag{1.6}$$

$$\Psi_{\boldsymbol{A}_{(i)}} = S_{T_{X(i)}} = \sum_{j=1}^{m} \left\| \boldsymbol{X}_{j(i)} - \overline{\boldsymbol{X}_{(\iota)}} \right\|_F^2 \tag{1.7}$$

where $\boldsymbol{A}_i$ and $\boldsymbol{A}_{i(n)}$ denote the i-th tensor sample and its mode-n unfolding respectively and $\overline{\boldsymbol{A}_{(n)}}$ denotes the mean mode-n unfolded sample:

$$\overline{\boldsymbol{A}_{(n)}} = \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{A}_{i(n)} \tag{1.8}$$

Because $\|\boldsymbol{A}\|_F = \left\|\boldsymbol{A}_{(i)}\right\|_F$, (1.6) and (1.7) are equal.

For supervised approaches, we define the between-class and within-class scatter:

$$S_{B_X} = \sum_{c=1}^{k} m_c \|\overline{\boldsymbol{X}_c} - \overline{\boldsymbol{X}}\|_F^2 \tag{1.9}$$

$$S_{W_X} = \sum_{j=1}^{m} \left\| \boldsymbol{X}_j - \overline{\boldsymbol{X}_{c_j}} \right\|_F^2 \tag{1.10}$$

Where m_c denotes the number of samples in class c. $\overline{\boldsymbol{X}_c}$ and $\overline{\boldsymbol{X}_{c_\iota}}$ denote the mean of class c and the mean of the class of sample i respectively.

Tensor decompositions, just like their equivalents for matrices, "break down" tensors into simpler building blocks that may be easier to interpret, require less storage and can help save up computational resources.

The CP/PARAFAC decomposition expresses a n-th order tensor $\boldsymbol{X} \in R^{p_1 \times \ldots \times p_n}$ as a sum of rank-1 tensors:

$$\boldsymbol{X} = \sum_{i=1}^{r} a^{(1)} \circ a^{(2)} \cdots \circ a^{(n)} \tag{1.11}$$

If r in (1.11) is equal to the rank of tensor $\boldsymbol{X}$, this can be seen as the higher order extension rank decomposition. However, it has been proved [35] that computing the Tensor rank is an NP-Complete problem in the general case. The most well-known way of computing the CP/PARAFAC decomposition is with Alternating Least Squares (ALS). First, for a tensor $\boldsymbol{X}$ we formulate the objective as:

$$\widehat{\boldsymbol{X}} = min_{\widehat{\boldsymbol{X}}} \left\| \boldsymbol{X} - \widehat{\boldsymbol{X}} \right\|^2, \widehat{\boldsymbol{A}} = [\![A^{(1)}, A^{(2)}, \ldots, A^{(n)}]\!] \tag{1.12}$$

Where $A^{(i)} \in R^{p_i \times r}$ $\forall i = 1, \ldots n$. In (2) we have "stacked" the vectors corresponding to the first mode in matrix $A^{(1)}$, the vectors corresponding to the second mode in $A^{(2)}$ and so on. Then, after initialization, we gradually solve for each component matrix while fixing the rest until a stopping criterion is met (max iterations, convergence or "good enough" approximation). Nonetheless, ALS is always "in danger" of being stuck at local minima.

The Tucker decomposition expresses a tensor as another tensor multiplied by a matrix at each mode:

$$\boldsymbol{X} = \boldsymbol{G} \times_1 A^{(1)} \times_2 A^{(2)} \dots \times_n A^{(n)} = [\![\boldsymbol{G}; A^{(1)}, \dots, A^{(n)}]\!] \tag{1.13}$$

Where $A^{(i)} \in R^{p_i \times q_i}\ \forall i = 1, \dots n$ and $\boldsymbol{G} \in R^{p_x \times p_1 \times p_2 \dots \times p_n}$ is referred to as the core tensor. While an ALS approach is also applicable here, the Higher Order Singular Value Decomposition (HOSVD) [36] procedure can also be used to compute a Tucker decomposition. In short, we perform SVD on each mode-i unfolding of the given tensor and set matrix $A^{(i)}$ equal to the most significant singular vectors. Then, we compute the core tensor using:

$$\boldsymbol{G} = \boldsymbol{X} \times_1 A^{(1)^T} \times_2 A^{(2)^T} \dots \times_n A^{(n)^T} \tag{1.14}$$

We note the matricized version of (1.14) that will be used extensively later:

$$\boldsymbol{X}_{(k)} = A^{(k)} \boldsymbol{G}_{(k)} (A^{(n)} \otimes A^{(n-1)} \otimes \dots \otimes A^{(k-1)} \otimes A^{(k+1)} \otimes \dots \otimes A^{(1)})^T \tag{1.15}$$

[34] is a great resource regarding the aforementioned decompositions and their applications.

1.4 Thesis outline

In Section 2, literature and formulations related to ours are presented. In Section 3, we present our proposed least squares formulation for MPCA and DATER. Lastly, in Section 4, we experimentally evaluate our proposed formulation on an Image Denoising and a Face Recognition (Classification) task and present the relevant results.

2. RELATED WORK

In this section, we will present various formulation of the relevant tensor CA methods. For the following, assume that we are given m tensor samples with n modalities (i.e., each sample $\boldsymbol{X_j} \in R^{p_1 \times p_2 \ldots \times p_n}\ \forall j = 1, \ldots, m$). We use $\boldsymbol{X} \in R^{p_1 \times p_2 \ldots \times p_n \times m}$ as a more 'compact' notation for the entirety of samples, with the additional mode of $\boldsymbol{X}$ indexing the samples themselves.

2.1 MPCA

MPCA, originally proposed by Lu et al. [16], is an unsupervised method that aims to determine a set of n projection matrices $U^{(1)}, U^{(2)} \ldots, U^{(n)}, U^{(i)} \in R^{q_i \times p_i}, q_i < p_i, \forall i = 1, \ldots, n$, such that the features of the projected samples $\boldsymbol{Y} = \boldsymbol{X} \times_1 U^{(1)} \times_2 U^{(2)} \ldots \times_n U^{(n)} \times_{n+1} \boldsymbol{I_m}$ present as much variation as possible. Following the definition of scatter (1.4) This objective can be formulated as:

$$\{U^{(1)}, \ldots, U^{(n)}\} = arg\ max_{U^{(1)}, \ldots, U^{(n)}} \sum_{j=1}^{m} \left\| \boldsymbol{Y_j} - \overline{\boldsymbol{Y}} \right\|_F^2$$

$$s.t\ U^{(i)^T} U^{(i)} = I_{p_i} \forall i = 1, \ldots, n \tag{2.1}$$

with $\boldsymbol{Y_j}$ denoting the j-th sample and $\overline{\boldsymbol{Y}} = \frac{\boldsymbol{1}}{\boldsymbol{m}} \sum_{j=1}^{m} \boldsymbol{Y_j}$ denoting the mean projected sample. (2.1) is known as the scatter maximization objective for MPCA. As the authors point out, replacing $\boldsymbol{Y_j}$ as $\boldsymbol{X_j} \times_1 U^{(1)} \times_2 U^{(2)} \ldots \times_n U^{(n)}$ and leveraging the fact that the data is centered[3] (both means $\overline{\boldsymbol{Y}}$ and $\overline{\boldsymbol{X}}$ are tensors of zeros), results in:

$$\{U^{(1)}, \ldots, U^{(n)}\} = arg\ max_{U^{(1)}, \ldots, U^{(n)}} \sum_{j=1}^{m} \left\| \boldsymbol{X_j} \times_1 U^{(1)} \times_2 U^{(2)} \ldots \times_n U^{(n)} \right\|_F^2 \tag{2.2}$$

Assume we are solving in an ALS fashion for $U^{(i)}$. This means all projection matrices $U^{(1)}, \ldots, U^{(i-1)}, U^{(i+1)}, \ldots, U^{(n)}$ are fixed and the unknown we are solving for is $U^{(i)}$. The constraint of (2.1) still applies. Using (1.15) and $\|A\|_F^2 = tr(AA^T)$:

$$\{U^{(i)}\} = arg\ max_{U^{(i)}} \sum_{j=1}^{m} tr\left(\left(U^{(i)} X_{j(i)} U_{\Phi_i}^T \right) \left(U^{(i)} X_{j(i)} U_{\Phi_i}^T \right)^T \right) \tag{2.3}$$

$$\{U^{(i)}\} = arg\ max_{U^{(i)}} \sum_{j=1}^{m} tr(U^{(i)} \underbrace{X_{j(i)} U_{\Phi^{(i)}}^T U_{\Phi^{(i)}} X_{j(i)}^T} U^{(i)^T}) \tag{2.4}$$

Where $U_{\Phi^{(i)}} = U^{(n)} \otimes \ldots \otimes U^{(i-1)} \otimes U^{(i-1)} \otimes \ldots U^{(1)}$. The underlined product can be interpreted as the i-mode scatter (1.7) of the data after being multiplied in each mode with the respective matrix except the i-th. This is referred to as a partial projection. We can solve (2.4) by setting $U^{(i)}$ equal to the q_i eigenvectors corresponding to the most significant eigenvalues of $X_{j(i)} U_{\Phi^{(i)}}^T U_{\Phi^{(i)}} X_{j(i)}^T$.

The algorithm to compute the relevant multilinear transformation receives the data and the max number of iterations as input. Then, in an iterative fashion the projection matrices are updated by finding the eigenvalues of the respective scatter matrix. The pseudocode of the method can be found at Algorithm 1.

[3] We have subtracted the mean sample from all samples of the train/input set.

Algorithm 1: MPCA

```
INPUT: Tensor samples $\boldsymbol{X_j} \in R^{p_1 \times p_2 \dots \times p_n}$ $\forall j = 1, \dots, m$, $\hat{k}$ (max iterations)
OUTPUT: MPCA Projection matrices $U^{(1)}, \dots, U^{(n)}, U^{(i)} \in R^{q_i \times p_i}, q_i < p_i, \forall i = 1, \dots, n$
BEGIN
for $\boldsymbol{j = 1 : m}$ do
  $\boldsymbol{X_j} \leftarrow \boldsymbol{X_j} - mean(\boldsymbol{X})$ # Center input data
Initialize $\boldsymbol{U^{(1)}}, \dots, \boldsymbol{U^{(n)}}$ # Initialize projection matrices
for $\boldsymbol{iter = 1 : \hat{k}}$ do
    for $\boldsymbol{i = 1 : n}$ do
        $\widehat{\boldsymbol{Y^{(i)}}} \leftarrow \boldsymbol{X} \times_1 U^{(1)} \times_2 U^{(2)} \dots \times_{i-1} U^{(i-1)} \times_{i+1} U^{(i+1)} \dots \times_n U^{(n)}$ # Partial projection
        $\widehat{S^{(i)}_{Y_{(i)}}} \leftarrow \sum_{j=1}^{m} X_{j(i)} U^{T}_{\Phi^{(i)}} U_{\Phi^{(i)}} X^{T}_{j(i)}$ # Compute $\boldsymbol{i}$-mode partial scatter
        $U^{(i)} \leftarrow q_i$ eigenvectors of most significant eigenvalues of $\widehat{S^{(i)}_{Y_{(i)}}}$ . # Update
return $U^{(1)}, \dots, U^{(n)}$
END
```

The authors of [16] claim that in each iteration, the objective function is non-decreasing and additionally it is upper bound by the scatter of the input data. As a result, good convergence behavior of the algorithm can be expected.

An additional issue to bear in mind is the initialization method for the projection matrices. Specifically, in [16], the following approaches are presented:

- *Random initialization* generates the elements of the projection matrices following a gaussian distribution in with $\mu = 0$ and $\sigma = 0.5$, followed by normalization[4].
- *Pseudo-identity* initialization truncates the last $p_i - q_i$ columns of the identity matrix of size p_i.
- *Full Projection Truncation* (FPT) performs HOSVD and initializes each matrix using the chosen number of components.

The first two methods are easier to implement, but that latter expresses better results regarding convergence and its error can be tightly bound [16].

One additional issue that needs to be addressed is selecting the number of components in each mode (i.e., $p_i \; \forall i = 1, \dots, n$). As the number of modalities (and features) grow, exhaustive search becomes prohibitive. One approach would be to incorporate the following ratio constraint in (2.1) or (2.6):

$$\frac{\prod_{i=1}^{n} q_i}{\prod_{i=1}^{n} p_i} < \Omega \tag{2.7}$$

[4] Each resulting matrix should have unit norm.

with Ω being the user-defined ratio of required dimensionality reduction. Lu et. al. [16] propose the Sequential Mode Truncation (SMT) method. In short, this method starts with $p_k = q_k, \forall i = 1, \dots, n$ and computes the amount of scatter lost if we discard the last feature of each mode. Then it discards the one with the minimum scatter loss and repeats this procedure until a specific scatter threshold is met. The Q-based method [16], on the other hand, is more straightforward: For each mode, truncate features as long a scatter requirement is met.

2.2 DATER

DATER, first published by Yan et al. [22], is a supervised method for Tensor Component Analysis and can be seen as tensor extension for LDA. DATER aims to maximize the ratio of between-class variation (scatter) while minimizing the within-class variation (scatter). As an intuition, we want samples of the same class to be projected as close as possible, while the sets of the classes should be distant and easily distinguishable. Assuming we have k classes with $n_c \; \forall c = 1, \dots, k$ samples each. DATER aims to find projection matrices $U^{(1)}, \dots, U^{(n)}$ that satisfy the above criteria:

$$\{U^{(1)}, \dots, U^{(n)}\} = arg\ max_{U^{(1)}, \dots, U^{(n)}} \frac{\sum_{c=1}^{k} n_c \|\overline{\boldsymbol{Y_c}} - \overline{\boldsymbol{Y}}\|_F^2}{\sum_{j=1}^{m} \left\| \boldsymbol{Y_j} - \overline{\boldsymbol{Y_{c_j}}} \right\|_F^2} \tag{2.8}$$

where $\overline{\boldsymbol{Y_c}}$ denotes the mean of samples in class c, $\overline{\boldsymbol{Y_{c_j}}}$ denotes the mean of the class that sample j belongs to and $\overline{\boldsymbol{Y}}$ denotes the mean of the data. If we use an ALS approach in this problem, fixing all projection matrices except the i-th then we obtain the following optimization problem from (2.8):

$$\{U^{(i)}\} = argmax_{U^{(i)}} \frac{tr\left({U^{(i)}}^T S_{B_y}^{(i)} U^{(i)} \right)}{tr\left({U^{(i)}}^T S_{W_y}^{(i)} U^{(i)} \right)} \tag{2.9}$$

where $S_{B_y}^{(i)}$ and $S_{W_y}^{(i)}$ denote the between-class and within-class scatter of the partially projected data respectively:

$$S_{B_{y(i)}}^{(i)} = \sum_{c=1}^{k} n_c \left\| \overline{\widetilde{Y_{c(\iota)}^{(\iota)}}} - \overline{\widetilde{Y_{(\iota)}^{(\iota)}}} \right\|_F^2 \tag{2.10}$$

$$S_{W_{y(i)}}^{(i)} = \sum_{j=1}^{m} \left\| \widetilde{Y_{J(\iota)}^{(\iota)}} - \overline{\overline{\widetilde{Y_{c_J(\iota)}^{(\iota)}}}} \right\|_F^2 \tag{2.11}$$

where $\overline{\widetilde{Y_{c(\iota)}^{(\iota)}}}$ denotes the mode-i unfolding of the c-th class mean of the partially projected data (all modes except the i-th), $\overline{\widetilde{Y_{(\iota)}^{(\iota)}}}$ denotes the mode-i unfolding mean of all the partially projected input data, $\widetilde{Y_{J(\iota)}^{(\iota)}}$ denotes mode-i unfolding of j-th partially projected sample and $\overline{\overline{\widetilde{Y_{c_J(\iota)}^{(\iota)}}}}$ denotes the mode-n unfolding mean of the class that the j-th is assigned. The solution to (2.8) is given by solving the following generalized eigenproblem:

$$S_{B_y}^{(i)} u = \lambda S_{W_y}^{(i)} u \tag{2.12}$$

Algorithm 2: DATER

```
INPUT: Tensor samples $\boldsymbol{X_i} \in R^{p_1 \times p_2 \dots \times p_n}\ \forall i = 1, \dots, m$, $\hat{k}$ (max iterations) and class labels $c \in \mathrm{R}^{\mathrm{m}}$
OUTPUT: DATER Projection matrices $U^{(1)}, \dots, U^{(n)}, U^{(i)} \in R^{p_i \times q_i}, q_i < p_i, \forall i = 1, \dots, n$
BEGIN
Initialize $U^{(1)}, U^{(2)} \dots, U^{(n)}$ # Initialize projection matrices
for $\boldsymbol{iter = 1{:}\hat{k}}$ do
    for $\boldsymbol{i = 1{:}n}$ do
        $\widehat{\boldsymbol{Y}^{(i)}} \leftarrow \boldsymbol{X} \times_1 U^{(1)} \times_2 U^{(2)} \dots \times_{i-1} U^{(i-1)} \times_{i+1} U^{(i+1)} \dots \times_n U^{(n)}$ # Partial projection
        $S^{(i)}_{B_{y(i)}} \leftarrow \sum_{c=1}^{k} n_c \left\| \overline{\widetilde{Y^{(i)}_{c(i)}}} - \overline{\widetilde{Y^{(i)}_{(i)}}} \right\|_F^2$ # Compute $\boldsymbol{i}$-mode between-class scatter
        $S^{(i)}_{W_{y(i)}} \leftarrow \sum_{j=1}^{m} \left\| \widetilde{Y^{(i)}_{j(i)}} - \overline{\widetilde{Y^{(i)}_{c_j(i)}}} \right\|_F^2$ # Compute $\boldsymbol{i}$-mode within-class scatter
        $U^{(i)} \leftarrow q_i$ eigenvectors of most significant eigenvalues of $(2.12)$ # Update
return $U^{(1)}, \dots, U^{(n)}$
END
```

and setting $U^{(i)}$ equal to the q_i generalized eigenvectors corresponding to largest eigenvalues. This procedure iterates over all of the modalities. The pseudocode for this method is presented in Algorithm 2. It has been shown that even for the simple case of 2 modalities, DATER may fail to converge [37]. Thus, stopping criteria such as maximum iterations or little change in projection matrices are invoked. Additionally, either of the initialization methods used for MPCA is also applicable here.

One key limitation of LDA was the fact that it could only produce at most $k-1$ features, where k is the number of classes. The authors of [22] prove that the maximum number of features (per mode) for DATER is:

$$min\{p_i, (k-1) \prod_{i \neq l} p_l\} \tag{2.13}$$

2.3 A least squares framework for component analysis

De la Torre [38] proposed a unified parametric objective function that can be used to formulate CA methods:

$$E(A, B) = \|W_r(\Gamma - BA^T Y)W_c\|_F^2 \tag{2.14}$$

W_r and W_c can be used as weight matrices for various methods that incorporate weight (e.g., weighted PCA). Γ and Y are commonly associated with the input or may be the input data matrix themselves. The matrix product BA^T is the transformation we are solving for. Depending on the values we chose for the parameters of (3.1) we can form different objectives. Differentiating with respect to A and B yields:

$$\frac{\partial \mathrm{E}}{\partial \mathrm{A}} = 2\mathrm{W}_{\mathrm{r}}^{2}\mathrm{B}\mathrm{A}^{\mathrm{T}}\mathrm{Y}\mathrm{W}_{c}^{2}\mathrm{Y}^{\mathrm{T}}\mathrm{A} - 2\mathrm{W}_{\mathrm{r}}^{2}\Gamma\mathrm{W}_{c}^{2}\mathrm{Y}^{\mathrm{T}}\mathrm{A} \tag{2.15}$$

$$\frac{\partial E}{\partial B} = 2YW_{c}^{2}\Gamma^{T}W_{r}^{2}B - 2YW_{c}^{2}Y^{T}AB^{T}W_{r}^{2}B \tag{2.16}$$

A way to minimize the objective (2.14) is to find minima by setting both derivatives equal to zero. If we do this for (2.15) the optimal B is given by:

$$B = \Gamma\mathrm{W}_{c}^{2}\mathrm{Y}^{\mathrm{T}}\mathrm{A}(\mathrm{A}^{\mathrm{T}}\mathrm{Y}\mathrm{W}_{c}^{2}\mathrm{Y}^{\mathrm{T}}\mathrm{A})^{-1} \tag{2.17}$$

Viewing (2.14) as a minimization problem and substituting B with the right side of (2.17) results in the following maximization objective:

$$E(A) = tr\big((A^{T}YW_{c}^{2}Y^{T}A)^{-1}(A^{T}YW_{c}^{2}\Gamma^{T}W_{r}^{2}\Gamma W_{c}^{2}Y^{T}A)\big) \tag{2.18}$$

which is an eigenproblem similar to (2.9). Similarly, we can do this for (2.16):

$$E(B) = tr\big((B^{T}W_{r}^{2}B)^{-1}(B^{T}W_{r}^{2}\Gamma W_{c}^{2}Y^{T}(YW_{c}^{2}Y^{T})^{-1}YW_{c}^{2}\Gamma^{T}W_{r}^{2}B)\big) \tag{2.19}$$

Typical eigensolvers may be unable to solve this problem due to ill-conditioning, stemming from the SSS problem or from receiving highly correlated input. Therefore, the author of this work proposes efficient numerical methods to tackle this difficulty. A straightforward approach would be to solve for A and B in an ALS fashion:

$$A^{(k+1)} = (Y\mathrm{W}_{c}^{2}Y^{T})^{-1}Y\mathrm{W}_{c}^{2}\Gamma^{T}\mathrm{W}_{r}^{2}B^{(k)}\left({B^{(k)}}^{T}\mathrm{W}_{\mathrm{r}}^{2}B^{(k)}\right)^{-1} \tag{2.20}$$

$$B^{(k+1)} = \Gamma\mathrm{W}_{c}^{2}Y^{T}A^{(k+1)}\left({A^{(k+1)}}^{T}Y\mathrm{W}_{c}^{2}Y^{T}A^{(k+1)}\right)^{-1} \tag{2.21}$$

Alternatively, subspace iteration [39, 40] is a numerically stable option that can be very fast with the appropriate initialization. Lastly, the author mentions that Gradient Descent-based and second order approaches are also applicable. For instance, a simple Gradient Descent approach would be utilizing the following:

$$A^{(k+1)} = A^{(k)} - \eta_{A}\frac{\partial \mathrm{E}\big(\mathrm{A}^{(\mathrm{k})}\big)}{\partial \mathrm{A}} \tag{2.22}$$

$$B^{(k+1)} = B^{(k)} - \eta_{B}\frac{\partial \mathrm{E}\big(\mathrm{B}^{(\mathrm{k})}\big)}{\partial \mathrm{B}} \tag{2.23}$$

where η_B and η_B denote the learning rate parameter for each factor matrix. Assuming our data is given in a vectorized format in a 2D matrix X, Table 2 sums up how relevant linear methods are formulated according to this framework.

Formulating CA methods according to this framework is beneficial. To begin with, having a common objective makes the similarities and differences between the methods transparent. Moreover, such formulation may improve the numerical stability when solving for the appropriate projection. That is because the LS Regression environment is well-studied and many efficient and numerically stable methods are applicable, such as those mentioned above. In addition, the formulation proposed may be the foundation for the development of novel CA methods or improvements to existing.

Table 2: Formulating PCA and LDA according to [38].

Method	W_r/W_c	Γ	Y	$Equation\ (3.5)$
PCA	I/I	X	I	$E_{PCA}(A) = tr\big((A^TA)^{-1}(A^TX^TXA)\big)$
LDA[5]	$(GG^T)^{-\frac{1}{2}}/I$	G^T	X	$E_{LDA}(A) = tr\big((A^TXX^TA)^{-1}(A^TXG(G^TG)^{-1}G^TX^TA)\big)$

De La Torre’s work was an inspiration to ours. Like the formulation proposed by [38], Least Square Regression is incorporated to our work. Even further, as will be discussed in Section 4, a Gradient Descent scheme, such as that presented in this section, was used for implementation. However, De La Torre’s work includes many other methods and even non-linear graph-embedding ones, whereas ours is solely based on (multilinear extensions of) PCA and LDA. Albeit mentioning tensors in the last Section, TCA methods are not touched upon further.

2.4 The graph embedding framework

Yan et al. [30] contributions are two-fold. First, their work presents a common way to formulate and solve for the parameters of various CA methods. According to the authors, input data shall be modeled in two weighted graphs: the “intrinsic” graph, which captures the similarity between data samples and the “penalty” graph, which describes the appropriate constraints. These graphs are defined by their adjacency matrices (W and B respectively) or their Laplacians[6], depending on which method is formulated. All methods culminate in the following generalized eigenvalue problem:

$$\tilde{L}u = \lambda\tilde{B}u \quad (2.24)$$

$\tilde{L}$ and $\tilde{B}$ are chosen with regard to the method. The solution is obtained by keeping the appropriate number of eigenvectors corresponding to the most significant eigenvalues. The authors also explain how this formulation can be extended in the non-linear and multi-modal case as well. Table 3 presents the relevant methods accordingly. For linear methods, assume we are given the input in matrix $X \in R^{d\times m}$ where m denotes the number of samples. For Tensor methods assume we are given m tensor samples with n modalities and we have grouped them, in a single tensor $\boldsymbol{X} \in R^{p_1\times p_2 \cdots \times p_n \times m}$, similarly to Section 2. Furthermore, this work presents “Marginal Fisher Analysis”, an LDA-based method for dimensionality reduction which allows more than $k-1$ projected features, where k denotes the number of classes.

A few clarifications regarding Table 3 follow. Columns W and B denote the adjacency matrices for the intrinsic and penalty graph, respectively. Columns $\tilde{L}$ and $\tilde{B}$ denote the matrices that will be used in (2.24) to solve this problem. Moreover, we kindly remind readers that the adjacency matrix of the constraint graph for LDA and DATER is known

[5] We will denote as $G \in R^{m\times k}$ (m: number of samples, k: number of classes) the indicator matrix with entries following:

$$G_{ij} = \begin{cases} 1, & c_i = j \\ 0, & else \end{cases}$$

[6] For a graph with adjacency matrix W, the graph Laplacian is defined as: $L = D - W, \mathrm{D_{ij}} = \begin{cases} \sum_{j\neq i} W_{ij}, i = j \\ 0,\ else \end{cases}$

as the centering matrix. The product XB performs centering of the data, while XBX^T is the total scatter tensor or, for 1D/vector data, the covariance matrix.

Table 3: Formulating relevant methods to our work according to [30].

Method	W	B	$\tilde{L}$	$\tilde{B}$
PCA[7]	$W_{ij} = \begin{cases} \frac{1}{m}, i \neq j \\ 0, i = j \end{cases}$	I	$\frac{1}{m} X \left(I - \frac{1}{m} 1_m 1_m^T \right) X^T$	XBX^T
MPCA (i-th mode, $X = \widehat{\boldsymbol{X}_{(\boldsymbol{n+1})}^{(\boldsymbol{\iota})}}$)	$W_{ij} = \begin{cases} \frac{1}{m}, i \neq j \\ 0, i = j \end{cases}$	I	$\frac{1}{m} \boldsymbol{X} \left(I - \frac{1}{m} 1_m 1_m^T \right) \boldsymbol{X}$	XBX^T
LDA[8]	$W_{ij} = \begin{cases} \frac{1}{n_{c_i}}, c_i = j \\ 0, else \end{cases}$	$I - \frac{1}{m} 1_m 1_m^T$	$X \left(I - \sum_{c=1}^{k} \frac{1}{n_c} e_c e_c^T \right) X^T$	XBX^T
DATER (i-th mode, $X = \widehat{\boldsymbol{X}_{(\boldsymbol{n+1})}^{(\boldsymbol{\iota})}}$)	$W_{ij} = \begin{cases} \frac{1}{n_{c_i}}, c_i = j \\ 0, else \end{cases}$	$I - \frac{1}{m} 1_m 1_m^T$	$\boldsymbol{X} \left(I - \sum_{c=1}^{k} \frac{1}{n_c} e_c e_c^T \right) X^T$	$\boldsymbol{X}BX^T$

Furthermore, we denote with $\widehat{\boldsymbol{X}_{(\boldsymbol{n+1})}^{(\boldsymbol{\iota})}}$ the partially projected data to all modes except the i-th, unfolded at the last mode, which indexes the samples. According to this and the definition of unfolding given at section 1.3, $\widehat{\boldsymbol{X}_{(\boldsymbol{n+1})}^{(\boldsymbol{\iota})}} \in \boldsymbol{R}^{\boldsymbol{m} \times \prod_{\boldsymbol{j}=1}^{\boldsymbol{n}} \boldsymbol{p_j}}$ and each row of it contains a sample in a vectorized form.

The graph-embedding framework provides a way to unify CA methods. To begin with, it provides an intuitive approach for interpretation of the input by modeling the data as weighted graphs. Moreover, the weights given to each graph can be used to compare and contrast CA methods. Parallel to this, such framework can be used in order to create new methods, even in the multilinear case.

The graph-embedding framework also provided us with another point of view for the methods of interest. The problems and their solutions were mathematically the same as those presented in 2.1 and 2.2. However, the matrices used in the eigenproblems had a different interpretation. While this is an equally valid approach, we state that numerical instability problems (e.g., SSS problem) often emerge when dealing with real-world tensor data [1]. In that case, eigensolvers may fail to converge and this may lead to problematic results. Our approach, albeit less interpretable, uses more efficient numerical procedures and manages to overcome such problems.

[7] 1_m denotes a vector of m ones. n_c denotes the number of samples belonging in class c.

[8] e^c denotes the m-dimensional vector of all zeros except and $e^c(j) = 1$ when the j-th sample belongs in the c-th class.

3. PROPOSED FORMULATION

In this section we will present our proposed formulation for MPCA and DATER. We begin this section with a review of LS Tensor Regression. Like before, assume we are given m tensor samples with n modalities, grouped in a single tensor $\boldsymbol{X} \in R^{p_1 \times p_2 \ldots \times p_n \times m}$ with $n+1$ modalities. The additional mode indexes the samples.

3.1 Least squares tensor Regression

In the Tensor Regression problem, we are given an input tensor $\boldsymbol{X} \in R^{p_1 \times p_2 \ldots \times p_n \times m}$ **,** the observed (scalar) output $y \in R$ and the bias $b \in R$ and our goal is to solve for the Tensor Regression coefficient $\boldsymbol{W} \in R^{p_1 \times p_2 \ldots \times p_n \times m}$ so:

$$y = < \boldsymbol{W}, \boldsymbol{X} > + b \tag{3.1}$$

One way to estimate $\boldsymbol{W}$ is solving the respective LS problem:

$$\boldsymbol{W} = argmin_{\boldsymbol{W}} \| y - < \boldsymbol{W}, \boldsymbol{X} > \|_F^2 \tag{3.2}$$

This model can be generalized for matrix or tensor output. For example, the simpler matrix case can be produced by replacing the inner product with the generalized inner product:

$$\begin{gathered} Y = < \boldsymbol{W}, \boldsymbol{X} >_n + \boldsymbol{B} \\ \boldsymbol{W} = argmin_{\boldsymbol{W}} \| Y - < \boldsymbol{W}, \boldsymbol{X} >_n \|_F^2 \end{gathered} \tag{3.3}$$

where $Y \in \mathrm{R}^{o \times m}, \boldsymbol{W} \in \mathrm{R}^{o \times p_1 \times p_2 \ldots \times p_n}$ and $B \in \mathrm{R}^{o \times m}$.

Several low-rank approaches suggest efficient solutions to the above problem. For example, Guo et al. [41] suggest replacing tensor $\boldsymbol{W}$ with its CP/PARAFAC decomposition. More specifically, the above objective can be formulated as:

$$\begin{gathered} Y = < \boldsymbol{W}, \boldsymbol{X} >_n + B \\ \{\boldsymbol{W}\} = arg\ min_{\boldsymbol{W}} \| Y - < \boldsymbol{W}, \boldsymbol{X} >_n \|_F^2 \\ s.t.\ \boldsymbol{W} = [\![V^{(0)}, V^{(1)}, \ldots, U^{(n)}]\!] \end{gathered} \tag{3.4}$$

where $V^{(0)} \in R^{r \times o}$ and $V^{(k)} \in R^{r \times p_k}\ \forall k = 1, \ldots, n$. Kossaifi et al. [42] propose a Tucker structure for $\boldsymbol{W}$. Accordingly, the above Tensor Regression problem can be formulated as:

$$\begin{gathered} Y = < \boldsymbol{W}, \boldsymbol{X} >_n + B \\ \{\boldsymbol{W}\} = arg\ min_{\boldsymbol{W}} \| Y - < \boldsymbol{W}, \boldsymbol{X} >_n \|_F^2 \\ s.t.\ \boldsymbol{W} = [\![\boldsymbol{G}; U^{(0)}, U^{(1)}, \ldots, U^{(n)}]\!] \end{gathered} \tag{3.5}$$

where $\boldsymbol{G} \in R^{q_0 \times q_1 \ldots \times q_n}, q_k < p_k\ \forall k = 0, \ldots, n, U^{(0)} \in \mathrm{R}^{q_0 \times o}$ and $U^{(k)} \in R^{q_k \times p_k}\ \forall k = 1, \ldots, n$. The key benefit of the low-rank formulations is the reduction of regression parameters (as can be seen on Table 4 below. However, we are more interested in formulation (3.5) because it offers the freedom of using a different number of components per mode.

Table 4: Number of parameters in Tensor Regression formulations.

Method	Full $\boldsymbol{W}$ (3.3)	CP/PARAFAC of $\boldsymbol{W}$ (3.4) [41]	Tucker of $\boldsymbol{W}$ (3.5) [42]
Regression parameters	$o \times \prod_{i=1}^{n} p_i$	$r \times o + \sum_{i=1}^{n} r \times p_i$	$\prod_{i=1}^{n} p_i + p_0 \times o + \sum_{i=1}^{n} q_i \times p_i$

3.2 MPCA as a LS Tensor Regression problem

In order to formulate MPCA as a LS Tensor Regression problem, firstly we need to express it as a minimization problem. Indeed, it can be proved that, just like its linear counterpart, MPCA can be formulated as minimization problem. The reconstruction error minimization objective for MPCA can be expressed as:

$$\{U^{(1)}, \dots, U^{(n)}\} = argmin_{U^{(1)},\dots,U^{(n)}} \sum_{j=1}^{m} \left\| \boldsymbol{X_j} - \boldsymbol{Y_j} \times_1 U^{(1)^T} \dots \times_n U^{(n)^T} \right\|_F^2$$

$$s.t\ U^{(i)} \in R^{q_i \times p_i}, U^{(i)^T} U^{(i)} = I_{p_i}, q_i < p_i, \forall \mathrm{i} = 1, \dots, \mathrm{n} \tag{3.6}$$

Substituting $\boldsymbol{Y_i}$ as $\boldsymbol{X_i} \times_1 U^{(1)} \times_2 U^{(2)} \dots \times_n U^{(n)}$ and using (1.2) gives us the reconstruction error minimization objective for MPCA:

$$\{U^{(1)}, \dots, U^{(n)}\} = arg\ min_{U^{(1)},\dots,U^{(n)}} \sum_{j=1}^{m} \left\| \boldsymbol{X_j} - \boldsymbol{X_j} \times_1 U^{(1)} U^{(1)^T} \dots \times_n U^{(n)} U^{(n)^T} \right\|_F^2$$

$$= arg\ min_{U^{(1)},\dots,U^{(n)}} \left\| \boldsymbol{X} - \boldsymbol{X} \times_1 U^{(1)} U^{(1)^T} \dots \times_n U^{(n)} U^{(n)^T} \right\|_F^2$$

$$s.t\ U^{(i)} \in R^{q_i \times p_i}, U^{(i)^T} U^{(i)} = I_{p_i}, q_i < p_i, \forall \mathrm{i} = 1, \dots, \mathrm{n} \tag{3.7}$$

We will now prove the equivalence of (2.1) and (3.6), following the methodology of Lu et al. [1]. For ease of presentation, we will use the more compact notation for both input and projected samples in (4.6):

$$\{U^{(1)}, \dots, U^{(n)}\} = argmax_{U^{(1)},\dots,U^{(n)}} \left\| \boldsymbol{X} - \boldsymbol{Y} \times_1 U^{(1)^T} \dots \times_n U^{(n)^T} \right\|_F^2 \tag{3.8}$$

Assume we are following an ALS approach and we are solving for the i-th projection matrix. Therefore, all the projection matrices except $U^{(i)}$ are known. Leveraging (1.15) and $\|A\|_F^2 = \left\| A_{(i)} \right\|_F^2$:

$$\{U^{(i)}\} = argmin_{U^{(i)}} \left\| X_{(i)} - U^{(i)^T} Y_{(i)} U_{\Phi_{(i)}} \right\|_F^2$$

$$U_{\Phi_{(i)}} = \left(U^{(n)} \otimes \dots \otimes U^{(i+1)} \otimes U^{(i-1)} \otimes \dots U^{(1)} \right) \tag{3.9}$$

Using $\|A\|_F^2 = tr(AA^T)$ results in:

$$\{U^{(i)}\} = argmin_{U^{(i)}} tr\left(\left(X_{(i)} - U^{(i)^T} Y_{(i)} U_{\Phi_{(i)}} \right) \left(X_{(i)} - U^{(i)^T} Y_{(i)} U_{\Phi_{(i)}} \right)^T \right)$$

$$= argmin_{U^{(i)}} tr\left(\left(X_{(i)} - U^{(i)^T} Y_{(i)} U_{\Phi_{(i)}} \right) \left({X_{(i)}}^T - U_{\Phi_{(i)}}^T {Y_{(i)}}^T U^{(i)} \right) \right) \tag{3.10}$$

It is true that $tr(ABCD) = tr(DABC)$ and $tr(A) = tr(A^T)$. Applying the distributive property in (4.11) yields:

$$U^{(i)} = argmin_{U^{(i)}} tr\left(X_{(i)} {X_{(i)}}^T\right) - 2tr\left(X_{(i)} U_{\Phi_{(i)}}^T {Y_{(i)}}^T U^{(i)}\right) + tr\left(Y_{(i)} {Y_{(i)}}^T\right) \tag{3.11}$$

Plugging in $\boldsymbol{Y_{(i)}} = U^{(i)} \boldsymbol{X_{(i)}} U_{\Phi_{(i)}}^T$:

$$\begin{aligned} \{U^{(i)}\} &= arg\ min_{U^{(i)}} tr(X_{(i)} {X_{(i)}}^T) - 2tr(Y_{(i)} {Y_{(i)}}^T) + tr(Y_{(i)} {Y_{(i)}}^T) \\ &= argmin_{U^{(i)}} tr\left(X_{(i)} {X_{(i)}}^T\right) - tr\left(Y_{(i)} {Y_{(i)}}^T\right) \end{aligned} \tag{3.12}$$

$\boldsymbol{X_{(i)}}$ refers to the unfolding of the input data that cannot be changed, therefore the objective is equal to:

$$\begin{aligned} \{U^{(i)}\} &= arg\ min_{U^{(i)}} - tr(Y_{(i)} {Y_{(i)}}^T) \\ &= arg\ max_{U^{(i)}} tr(Y_{(i)} {Y_{(i)}}^T) \\ &= arg\ max_{U^{(i)}} \|\boldsymbol{Y}\|_F^2 \\ &= argmax_{U^{(i)}} \sum_{i=1}^{N} \|\boldsymbol{Y_i} - \boldsymbol{\bar{Y}}\|_F^2 \end{aligned} \tag{3.13}$$

where the last equality holds because the data is centered and the mean of the samples (both input and projected) is zero. Notice that (3.13) is indeed the scatter maximization objective for MPCA, therefore the two objectives are interchangeable. ■

An intuition on the equality of the two objectives for linear PCA can be seen by reviewing Figure 1. Notice that the same black line (Principal Axis) that maximizes the variation of the projection “forces” the reconstruction error (red lines) to be minimum, according to Pythagorean Theorem. Assuming we are solving in an ALS fashion for the i-th projection matrices, (3.7) via (1.15) becomes:

$$\begin{gathered} \{U^{(i)}\} = argmin_{U^{(i)}} \sum_{j=1}^{m} \left\| X_{j(i)} - U^{(i)} {U^{(i)}}^T X_{j(i)} U_{\Phi_{(i)}}^T \right\|_F^2 \\ with\ U_{\Phi_{(i)}} = {U^{(n)}}^T U^{(n)} \otimes \ldots \otimes {U^{(i-1)}}^T U^{(i-1)} \otimes {U^{(i+1)}}^T U^{(i+1)} \otimes \ldots \otimes {U^{(1)}}^T U^{(1)} \\ s.t\ U^{(i)} \in R^{q_i \times p_i}, {U^{(i)}}^T\ U^{(i)} = I_{p_i}, q_i < p_i, \forall \mathrm{i} = 1, \ldots, \mathrm{n} \end{gathered} \tag{3.14}$$

Alternatively, if we group all the samples in tensor $\boldsymbol{X}$:

$$\begin{gathered} \{U^{(i)}\} = argmin_{U^{(i)}} \left\| X_{(i)} - U^{(i)} {U^{(i)}}^T X_{(i)}\ U_{\Phi_{(i)+}}^T \right\|_F^2 \\ with\ U_{\Phi_{(i)+}} = {I_{n+1}}^T I_{n+1} \otimes {U^{(n)}}^T U^{(n)} \otimes \ldots \otimes {U^{(i-1)}}^T U^{(i-1)} \otimes {U^{(i+1)}}^T U^{(i+1)} \otimes \ldots \otimes {U^{(1)}}^T U^{(1)} \\ s.t\ U^{(i)} \in R^{q_i \times p_i}, {U^{(i)}}^T\ U^{(i)} = I_{p_i}, q_i < p_i, \forall \mathrm{i} = 1, \ldots, \mathrm{n} \end{gathered} \tag{3.15}$$

Algorithm 3: LS-MPCA (GD)

```
INPUT: Tensor samples $\boldsymbol{X_i} \in R^{p_1 \times p_2 \ldots \times p_n}$ $\forall i = 1, \ldots, m$, $\hat{k}$ (max iterations)
OUTPUT: MPCA Projection matrices $U^{(1)}, \ldots, U^{(n)}, U^{(i)} \in R^{q_i \times p_i}, q_i < p_i, \forall i = 1, \ldots, n$
BEGIN
for $\boldsymbol{i = 1{:}m}$ do
  $\boldsymbol{X_i} \leftarrow \boldsymbol{X_i} - mean(\boldsymbol{X})$  # Center input data
Initialize $\boldsymbol{U^{(1)}}, \ldots, \boldsymbol{U^{(n)}}$ # Initialize projection matrices
for $\boldsymbol{iter = 1{:}\hat{k}}$ do
   for $\boldsymbol{i = 1{:}n}$ do
      $U^{(i)} \leftarrow U^{(i)} - \eta \frac{\partial \mathrm{E}(\mathrm{U}^{(\mathrm{i})})}{\partial U^{(i)}}$ # $E(U^{(i)})$ is the objective function of $(3.15)$ or $(3.16)$
   if little change in objective $\boldsymbol{E}$: break
return $\boldsymbol{U^{(1)}}, \ldots, \boldsymbol{U^{(n)}}$
END
```

We may now formulate Problem (4.7) as a LS Tensor Regression problem:

$$\boldsymbol{X} = <\boldsymbol{W}, \boldsymbol{X}>_n + \boldsymbol{B}$$
$$\{\boldsymbol{W}\} = argmin_{\boldsymbol{W}} \|\boldsymbol{X} - <\boldsymbol{W}, \boldsymbol{X}>_n\|_F^2$$
$$s.t. \boldsymbol{W} = [\![\boldsymbol{G}; V^{(1)}, V^{(2)}, \ldots, V^{(n+1)}]\!] \in \mathrm{R}^{p_1 p_2 \ldots p_n \times p_1 \times p_2 \ldots \times p_n},$$
$$V^{(l)} = \begin{cases} I, \ l = 1 \\ U^{(l-1)} U^{(l-1)^T}, 1 < l \leq n+1 \end{cases},$$
$$\boldsymbol{G}_{(1)} = I_{p_1 p_2 \ldots p_n}, U^{(i)} \in R^{q_i \times p_i}, U^{(i)^T} U^{(i)} = I_{p_i}, q_i < p_i, \forall \mathrm{i} = 1, \ldots, \mathrm{n} \qquad (3.16)$$

It straightforward to prove that the two objectives are equivalent by leveraging

$$<\boldsymbol{W}, \boldsymbol{X}>_n = W_{(1)} X_{(n+1)}^T \qquad (3.17)$$

and (1.15) ($\boldsymbol{W}$ and $\boldsymbol{X}$ as defined previously). Pseudocode on solving MPCA with Gradient Descent on the respective LS problem is shown in Algorithm 3.

3.3 DATER as a LS Regression problem

Before formulation DATER, we will briefly discuss the work of Ye [43], which will be used. The author of this work proposes a Least Squares regression formulation for LDA, so that more efficient and stable methods are applicable for solving, similarly to this work. Assuming we are given a data matrix $X \in R^{d\times m}$ and an indicator matrix $Y \in R^{k\times m}$, multiclass LDA can be expressed as finding $W \in R^{k\times d}$ (k denotes the number of classes):

$$W = argmin_W \left\| \ddot{Y} - WX^T \right\|_F^2, \ddot{Y} = \begin{cases} \sqrt{\frac{m}{n_j}} - \sqrt{\frac{n_j}{m}}, c_i = j \\ -\sqrt{\frac{n_j}{m}}, otherwise \end{cases} \tag{3.18}$$

where n_j denotes the number of samples belonging in class j and m is the total number of samples. Solving this LS problem yields the same solution as solving the eigenproblem formulation for LDA, such as the one presented in Table 3, under the mild condition:

$$rank(S_{T_X}) = rank(S_{B_X}) + rank(S_{W_X}) \tag{3.19}$$

which is usually true in high-dimensional datasets, according to this work.

We have stated that DATER can be seen as mode-wise LDA, therefore we may adopt this formulation to achieve our goal. Specifically, we may formulate DATER as:

$$\begin{gathered} \boldsymbol{Y} =< \boldsymbol{W}, \boldsymbol{X} >_n + \boldsymbol{B} \\ \{\boldsymbol{W}\} = argmin_{\boldsymbol{W}} \|\boldsymbol{Y} - < \boldsymbol{W}, \boldsymbol{X} >_n\|_F^2 \\ s.t. \boldsymbol{W} = [\![\boldsymbol{G}; V^{(1)}, V^{(2)}, \dots, V^{(n+1)}]\!] \in \mathrm{R}^{k\times p_1 \times p_2 \dots \times p_n}, \\ V^{(l)} = \begin{cases} I, \ l = 1 \\ U^{(l-1)}, \ 1 < l \leq n+1 \end{cases}, \\ Y = \ddot{Y}, U^{(i)} \in R^{q_i \times p_i}, q_i < p_i, \forall \mathrm{i} = 1, \dots, \mathrm{n} \end{gathered} \tag{3.20}$$

A more concise view on this objective can be seen by replacing $\boldsymbol{W}$ with its decomposition:

$$\begin{aligned} \{\boldsymbol{W}\} &= argmin_{\boldsymbol{W}} \|\boldsymbol{Y} - < \boldsymbol{W}, \boldsymbol{X} >_n\|_F^2 \\ &= argmin_{\boldsymbol{W}} \left\| \boldsymbol{Y} - \boldsymbol{W}_{(1)} \boldsymbol{X}_{(n+1)}^T \right\|_F^2 \\ &= argmin_{\boldsymbol{G}, U^{(1)}, \dots, U^{(n)}} \left\| \boldsymbol{Y} - \boldsymbol{G}_{(1)} (U^{(n)^T} \otimes \dots \otimes U^{(1)^T}) \boldsymbol{X}_{(n+1)}^T \right\|_F^2 \end{aligned} \tag{3.21}$$

One difference between (3.20) and (3.16) is the fact that the core tensor $\boldsymbol{G}$ does not have a fixed value in the current objective and has to be re-calculated in each iteration. The steps of a Gradient Descent approach to this problem are presented in Algorithm 4.

Algorithm 4: LS-DATER (GD)

INPUT: Tensor samples $\boldsymbol{X_i} \in R^{p_1 \times p_2 \dots \times p_n}$ $\forall i = 1, \dots, m$, $\hat{k}$ (max iterations) and class labels $c \in \mathrm{R}^{\mathrm{m}}$

OUTPUT: DATER Projection matrices $U^{(1)}, \dots, U^{(n)}, U^{(i)} \in R^{q_i \times p_i}, q_i < p_i, \forall i = 1, \dots, n$

```
BEGIN
Initialize $\boldsymbol{U^{(1)}}, \boldsymbol{U^{(2)}} \dots, \boldsymbol{U^{(n)}}$ # Initialize projection matrices
for $\boldsymbol{iter = 1 : \hat{k}}$ do
    for $\boldsymbol{i = 1 : n}$ do
        $U^{(i)} \leftarrow U^{(i)} - \eta \frac{\partial \mathrm{E}(\mathrm{U}^{(\mathrm{i})})}{\partial U^{(i)}}$ # $E(U^{(i)})$ is the objective function of $(3.20)$ or $(3.21)$
        $G_{(1)} \leftarrow G_{(1)} - \eta \frac{\partial \mathrm{E}(G_{(1)})}{\partial G_{(1)}}$ # $E(G_{(1)})$ is the objective function of $(3.20)$ or $(3.21)$
return $U^{(1)}, \dots, U^{(n)}$
END
```

4. EXPERIMENTAL EVALUATION

We implemented the methods mentioned in Section 2 and 4 and in this we evaluate their performance on real-world data. Specifically, we consider the image denoising task on the cropped Extended Yale Face Database B [44, 45] and the face recognition (classification) task on the ORL dataset [46]. NumPy [47], PyTorch [48] and Tensorly [49] were used extensively for representing and performing operations on the data. SciPy [50] and Python Image Library (PIL) [51] were also used for reading the data. Matplotlib [52] was used for plotting. For methods that required solving eigenproblems, the *eig* function[9] of [47] was used, while the *autograd* package[10] of [48] was used for differentiation in the implemented Gradient Descent approaches.

4.1 Image denoising

The Extended Yale Face Database B consists of 64 images of 38 people, summing to a total of 2414 images. For this experiment, we considered only the frontal pose (Pose 01) of each person with frontal illumination. Therefore, the data used contained a total of 38 images (one for each person).

4.1.1 Dividing the dataset

Our goal in this experiment is to compare the performance of the relevant methods on unseen data. We begin this experiment by dividing the available data in the train (30 images) and test set (8 images) at random. We further divided the train set in 3 folds (10 images in each) and added Gaussian noise (using scikit-image [53]) with deviations $\sigma_{test} = 0.007, 0.01, 0.02, 0.05$ to the test set. We performed Hyperparameter selection on the methods used (Gaussian Filter, Total Variation (TV) Filter [53, 54], Vectorization + Linear PCA [55], MPCA [16] and the (proposed) LS-MPCA by 3-fold Cross validation by adding gaussian noise on each validation test with deviations $\sigma_{valid} = 0.005, 0.008, 0.015, 0.03, 0.06$. Using the obtained best performing hyperparameters, we execute all algorithms on the test set. The results are validated with the following criteria (Assume we are given $X_{(original)}, X_{(denoised)} \in R^{n_j \times n_i}$):

- *Mean Squared Error (MSE):* A straightforward mathematical way to compute the difference between the denoised image and the original. MSE is given by:
$$MSE = \frac{1}{m}\sum_{j=1}^{n_j}\sum_{i=1}^{n_i}\left(X_{(original)ij} - X_{(denoised)ij}\right) \tag{4.1}$$
- *Peak Signal-to-Noise Ratio (PSNR):* A well-known reconstruction assessment metric that can be defined relatively to MSE:
$$PSNR = 10log_{10}\left(\frac{max^2\{X_{(original)}\}}{MSE}\right) \tag{4.2}$$
where $max\{X\}$ returns the largest possible value for X.
- *Absolute Relative Error (ARE):* is defined as:
$$ARE = \frac{\left\|X_{(original)} - X_{(denoised)}\right\|}{\left\|X_{(original)}\right\|} \tag{4.3}$$

[9] https://numpy.org/doc/stable/reference/generated/numpy.linalg.eig.html

[10] https://pytorch.org/docs/stable/autograd.html

- *Execution time* (s)
- *Qualitative assessment of resulting denoised images*

A high-level overview of this task is presented in Figure 5. Figure 6 presents visual results for each of the 8 images of the test set and Table 5 presents relevant metrics.

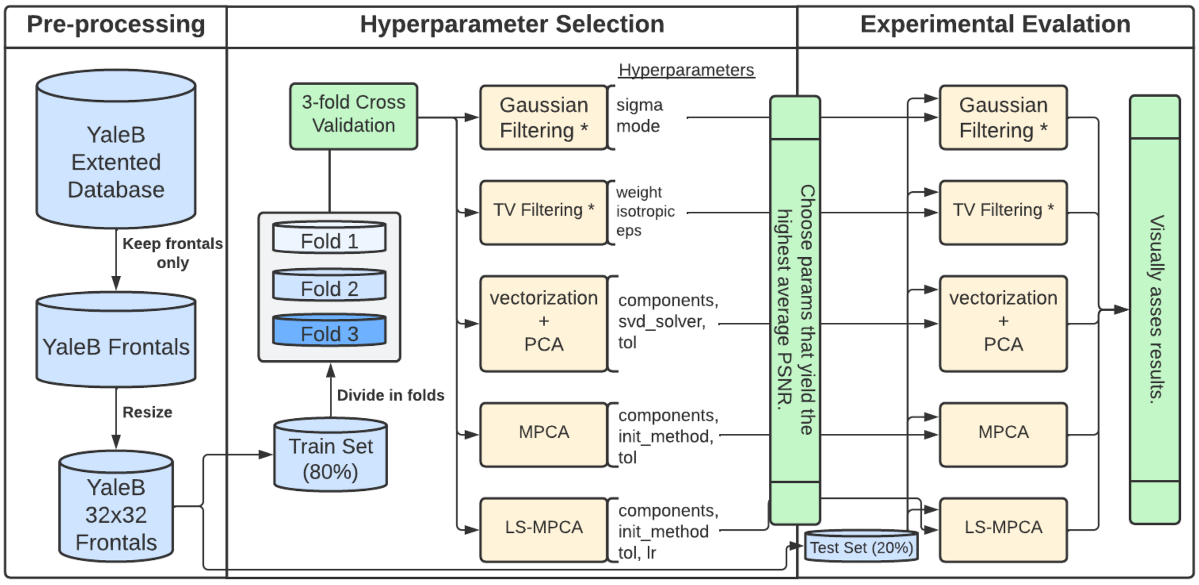

Figure 5: Overview of the Image Denoising experiment (Methods with * do not require any training)

Table 5: Numerical results of relevant methods on test set.

Method	Average PSNR	Average MSE	Average Time (s)	Average ARE
Gaussian Filtering	23.828	0.005	0.00019	0.132
TV Filtering	21.927	0.008	0.00011	0.169
PCA	22.388	0.007	0.00537	0.158
MPCA	22.520	0.007	0.36706	0.159
LS-MPCA	22.375	0.007	0.03149	0.160

Metric-wise, Gaussian Filtering seems to outperform the rest of the methods, yielding the highest PSNR and the lowest MSE and ARE. However, the denoised images produced by this method show blurriness, which makes the detection of edges by eye sometimes harder. PCA seems to produce the sharpest images, but some details on each face sample seem to be lost in the projection, which of course has a toll on the quality of the result. MPCA and LS-MPCA have similar behavior, with the effect of the denoising being especially visible at lower settings of variance of the noise. Lastly, regarding the execution time, we notice that Gaussian and TV Filtering perform best, which is to be expected as these methods are applied directly to the images without requiring any training. PCA performs the fastest out of the rest of the methods, and we note the advantage of LS-MPCA on execution time (one order of magnitude less) over the standard MPCA.

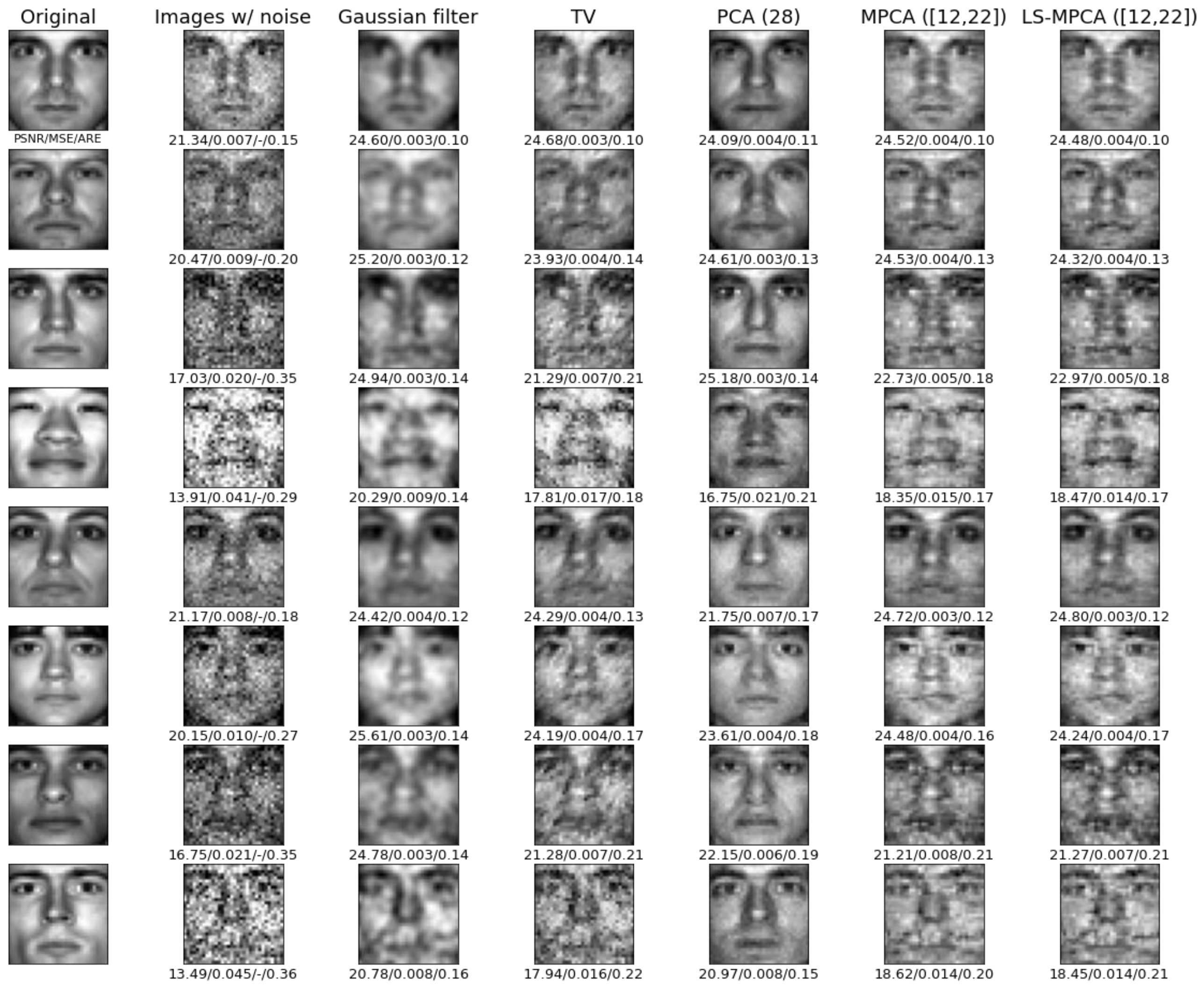

Figure 6: Visual results of implemented methods on the test set. The numbers in the parentheses denote number of components.

4.1.2 Execution on full set

In order to get another view at how each method performs, we added gaussian noise to all of the available data (38 images) with deviations $\sigma_{full} = 0.007, 0.01, 0.05, 0.075$ and used all of the methods to perform denoising. Like we did in section 5.1.1, we begin with a gridsearch approach to determine -among other parameters- the number of components for PCA-based methods. The results of this procedure are presented in Figures 7 and 8. Numerical results for all methods (with the best performing parameters) can be found on Table 6.

Similarly to the previous experiment, Gaussian Filtering seems to yield the best metrics. Regarding PCA, we observe that increasing the number of components used, results in deteriorating denoising quality. This seems also to be the case with MPCA and LS-MPCA, but we notice that using 4-6 componets in the first mode and at least 8 and no more than 12 components in the second results in rise of the relevant metrics. Methods that require no training, as expected, provide results more quickly. PCA uses less components, so less computations are generally required than its tensor extensions. As also noted in the previous experiment, LS-MPCA's execution time is less than that of MPCA, which can easily be seen at the legend-colorbar of Figures 7 and 8. In this experiment, we also measured the orthogonality of the produced projection matrices using the distance $\|UU^T - I\|_F^2$. Both MPCA and LS-MPCA presented similar behavior in this aspect.

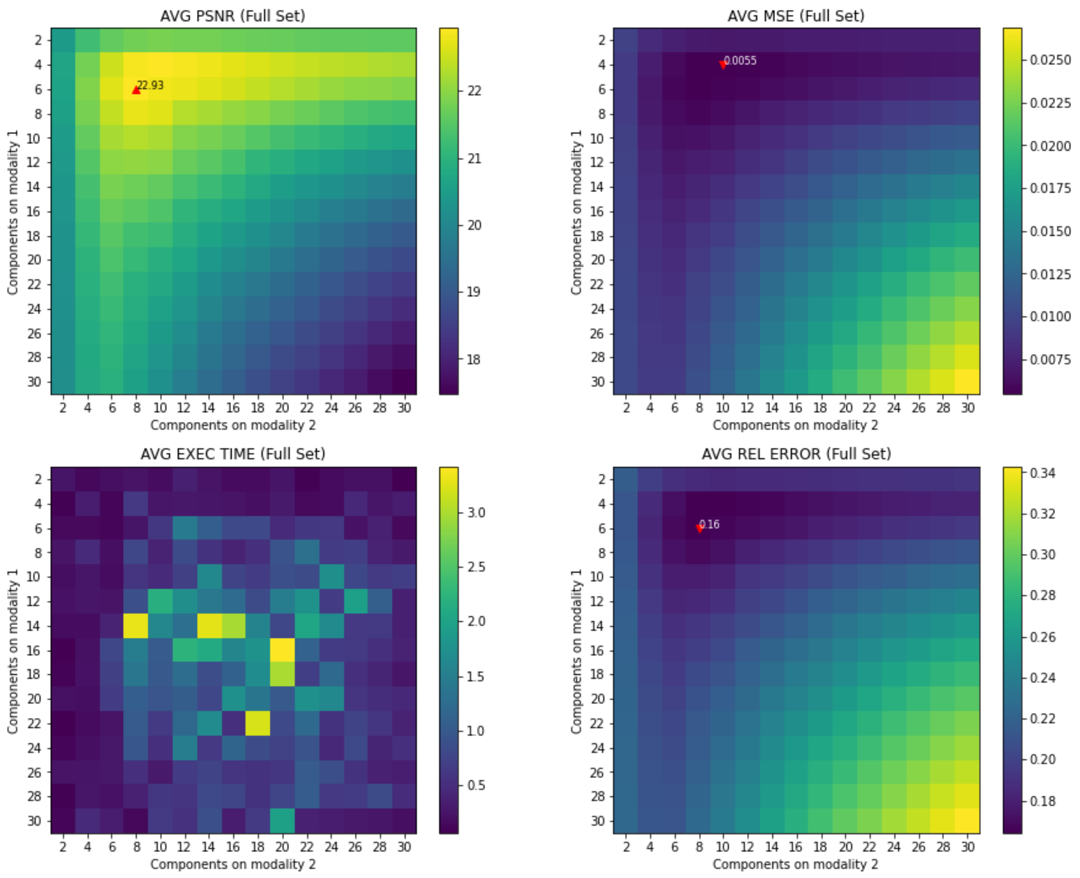

Figure 7: Numerical results of MPCA for various number of components per mode (Full Dataset). Red triangles denote the global maxima-minima.

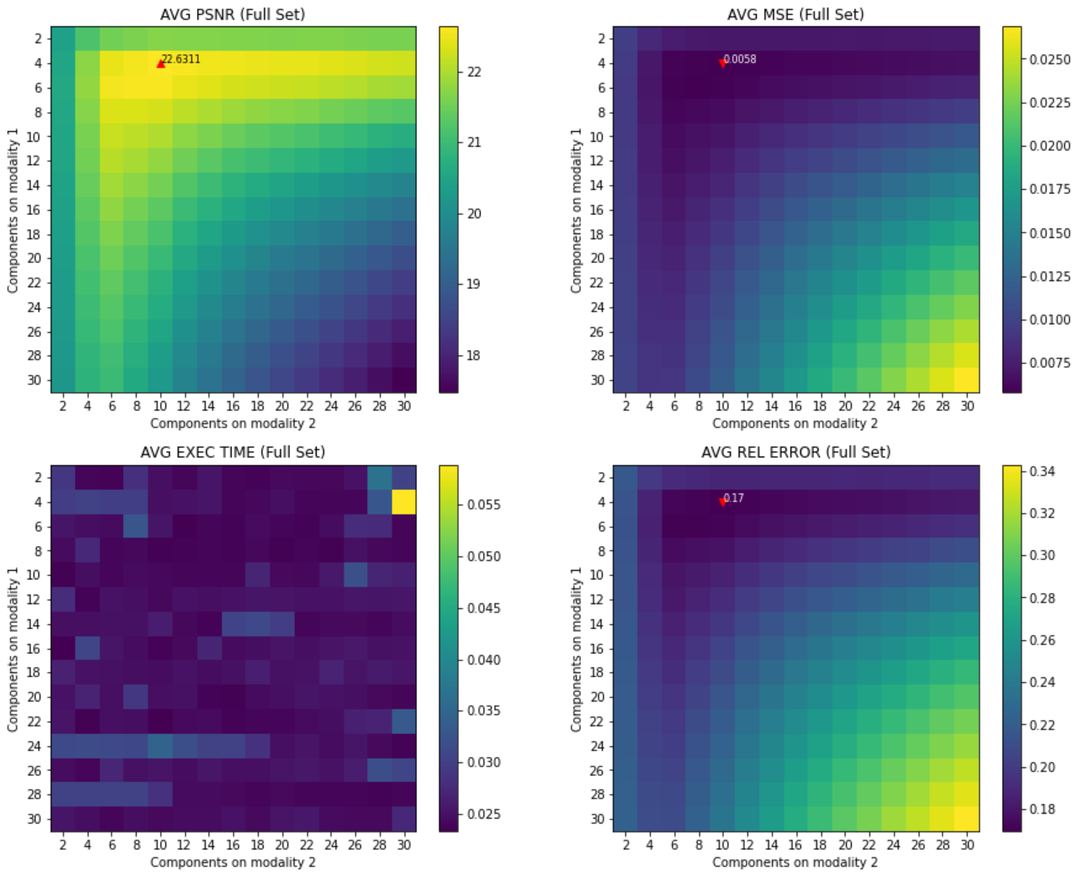

Figure 8: Numerical results of LS-MPCA for various number of components per mode (Full Dataset). Red triangles denote the global maxima-minima.

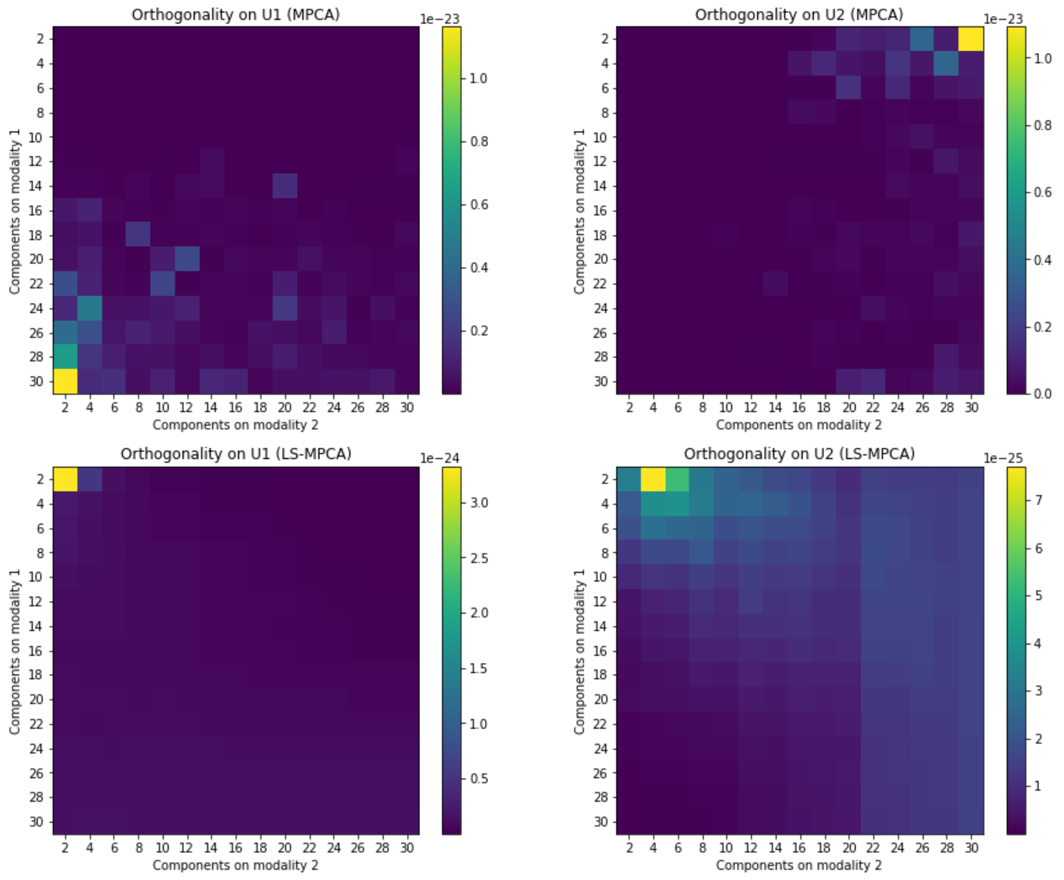

Figure 9: Orthogonality of computed projection matrices (Full Dataset).

Table 6: Results on Full Dataset denoising.
Bold values indicate best metrics of the respective method.

Method	Average PSNR	Average MSE	Average Time (s)	Average ARE
Gaussian Filtering	**23.505**	**0.005**	0.0002	**0.157**
TV ($w = 2.0$)	**22.469**	**0.007**	0.0002	**0.182**
TV ($w = 5.0$)	21.084	0.011	0.0002	0.222
PCA (2)	**20.392**	**0.01**	0.011	**0.22**
PCA (4)	19.861	0.011	0.012	0.236
PCA (6)	19.349	0.014	0.015	0.256
PCA (8)	18.896	0.017	0.014	0.275
PCA (10)	18.579	0.019	0.019	0.291
PCA (12)	18.255	0.022	0.02	0.306
PCA (14)	17.947	0.024	0.02	0.318

PCA (16)	17.651	0.026	0.008	0.331
PCA (18)	17.482	0.027	0.009	0.341
PCA (20)	17.6	0.028	0.009	0.343
PCA (22)	17.809	0.028	0.121	0.34
PCA (24)	17.891	0.028	0.025	0.339
PCA (26)	17.868	0.028	0.01	0.34
PCA (28)	17.825	0.028	0.011	0.34
PCA (30)	17.689	0.028	0.014	0.344
MPCA (10,10)	22.25	0.007	0.608	0.182
MPCA (12,12)	21.664	0.008	1.630	0.198
MPCA (15,15)	20.9	0.011	0.810	0.221
MPCA (20,20)	19.680	0.015	0.977	0.262
MPCA (25,25)	18.524	0.021	0.536	0.303
MPCA (30,30)	17.475	0.027	0.159	0.342
MPCA (6,8)	**22.934**	**0.005**	0.248	**0.164**
MPCA (4.10)	22.914	0.005	0.153	0.164
LS-MPCA (10,10)	22.007	0.007	0.027	0.186
LS-MPCA (12,12)	21.507	0.008	0.0238	0.2
LS-MPCA (15,15)	20.82	0.011	0.0238	0.221
LS-MPCA (20,20)	19.624	0.015	0.025	0.262
LS-MPCA (25,25)	18.572	0.0207	0.025	0.301
LS-MPCA (30,30)	17.474	0.027	0.024	0.342
LS-MPCA (4,10)	**22.631**	**0.006**	0.024	**0.17**

4.2 Face recognition

Our Database of Faces (ORL) [46] is dataset with 400 images of 40 subjects. The datasets contain multiple images of each subject, varying the light and facial expression. Our goal in this experiment would be to obtain the most discriminative features using LDA-based methods and assess their quality by using these features as input to baseline classifiers (Nearest Neighbor and Linear Support Vector Machines).

We begin this experiment by dividing the data in the train (300 samples) and test set (100 samples). Then, we perform stratified 5-fold Cross Validation to determine the best performing parameters for each method. The methods we will compare are vec+LDA (vectorization prior to applying LDA), DATER (Section 2), LS-DATER (proposed method at Section 4) and PCA+LDA (vectorization and PCA prior to LDA). An overview to this experiment is presented in Figure 10.

Having obtained the best performing hyperparameters, we present the results for each method in Figure 11. From the perspective of accuracy, we notice that all methods predict the correct label for more than 90% of the test samples, with DATER even reaching 100%. Nonetheless, all the methods have comparable results. By viewing the lower part of Figure 9, however, we see where the methods differ: their execution time. vec+LDA and PCA+LDA present the lowest times. Regarding DATER and LS-DATER, we once again notice the advantage the latter has when it comes to computation time over the former.

Figure 12 shows heatmaps of the orthogonality of each of the produced projection matrices for DATER and our proposed LS-DATER for any combination of components per mode. One can easily see that our proposed formulation is able to retain the orthogonality regardless the components used, whereas the DATER fails to do so when a increasing the number of components (especially in modality 1). We kindly remind readers that orthogonality is not a strict requirement for DATER. However, using LS-DATER in this case results in less correlated features.

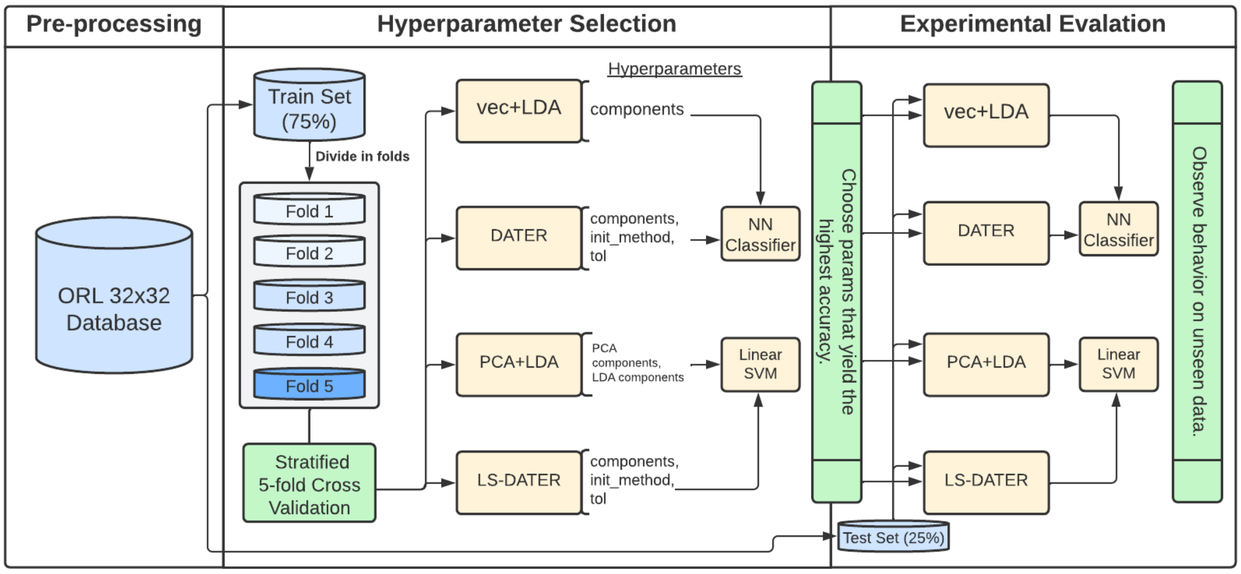

Figure 10: Overview of Face Recognition task.

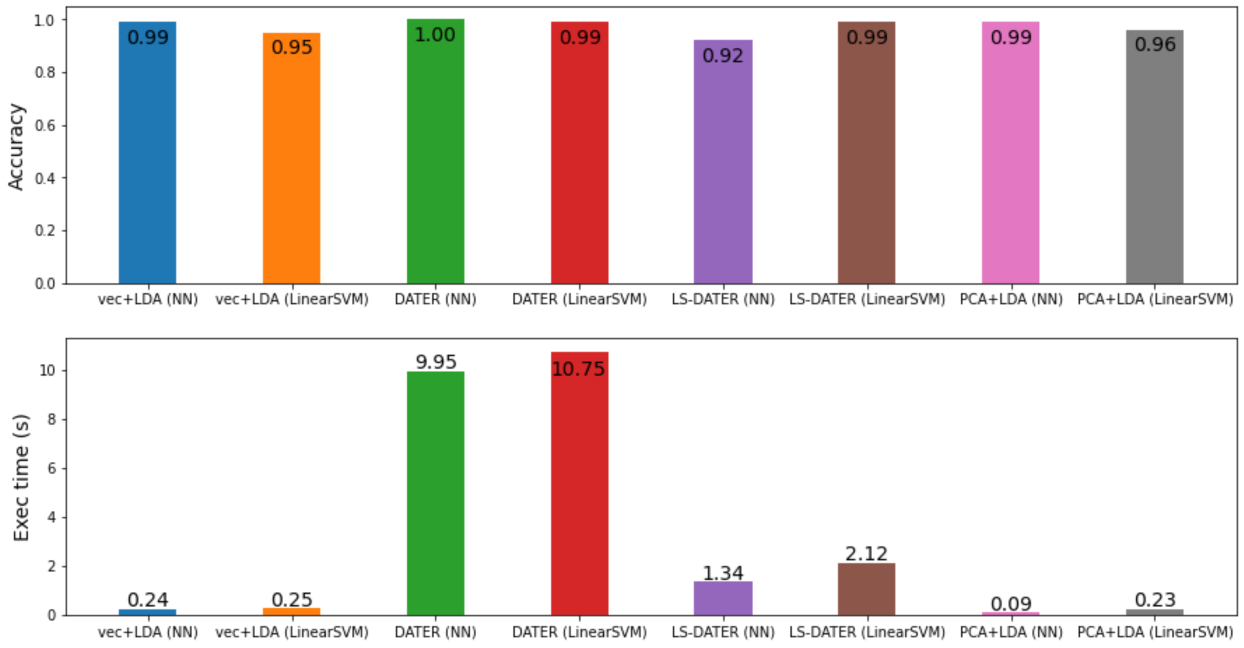

Figure 11: Classification results for Face Recognition Task (test set).

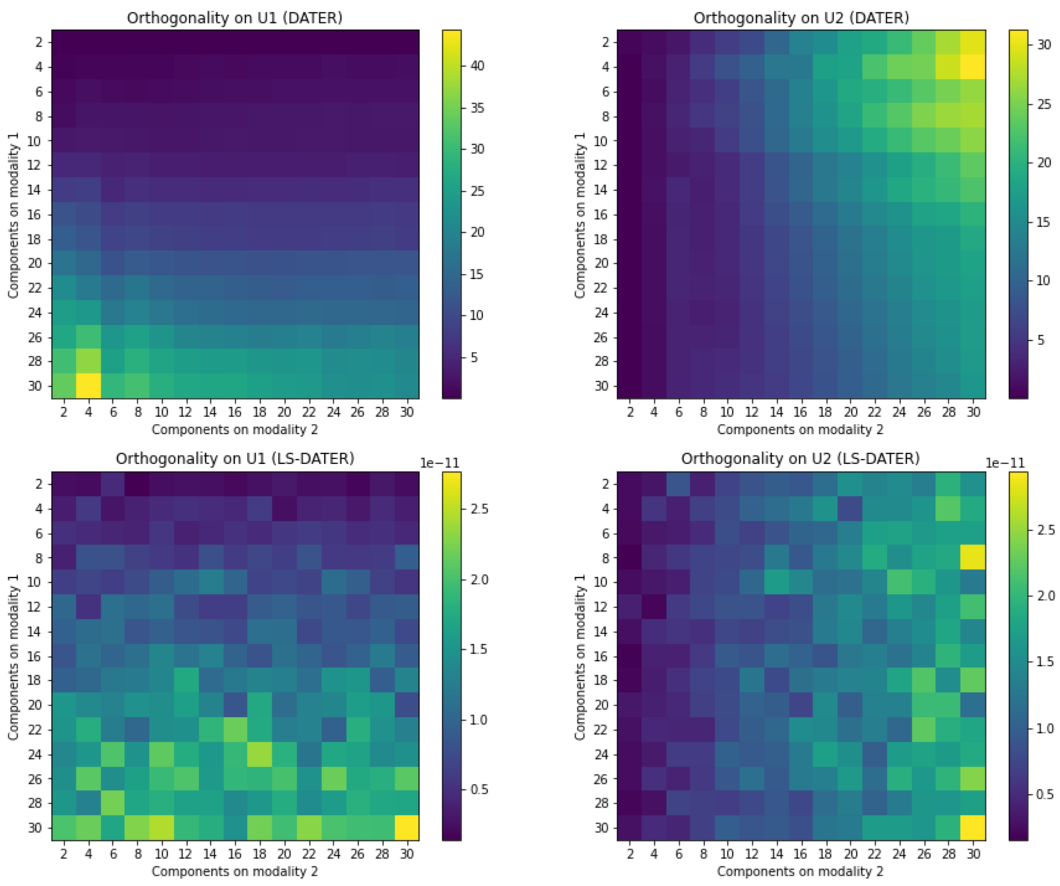

Figure 12: Orthogonality of produced projection matrices (Face Recognition)

5. CONCLUSION

In this thesis, we have presented various CA methods and their extensions for tensor data. More specifically, we have investigated MPCA and DATER and presented how these methods can be formulated as generalized eigenproblems. Such approach may lead to problematic results in quality because the involved matrices may be near-singular (e.g., highly correlated data). Additionally, tensor data is typically highly dimensional, and this may result in surge in the execution time of eigensolvers. As a remedy, we proposed a LS Tensor Regression formulation for MPCA and DATER, thus making more numerically stable and less computationally demanding methods usable. In the last section, we implemented and experimentally assessed our proposition and noticed competitive results. Moreover, the results of our formulation were similar to the eigen-formulation, but the former had the advantage of a smaller (by one order of magnitude) execution time. Furthermore, in the case of LS-DATER, the resulting mapping contained features that were more uncorrelated, which of course is an extra benefit.

As mentioned at the start of this work, PCA and LDA are the most well-known but not the only CA methods in the literature. Many of those methods have been extended to tensor data as well, such as MMICA [26] and MCCA [29]. For future work, we are planning to investigate formulations of more methods in the LS Tensor Regression setting. The end goal of this investigation would be to create a unified framework for Tensor CA methods.

ABBREVIATIONS – ACRONYMS

ALS	Alternating Least Squares
CCA	Canonical Correlation Analysis
DATER	Discriminant Analysis with tensor representation
fMRI	functional Magnetic Resonance Imaging
FPT	Full projection truncation
GTDA	General Tensor Discriminant Analysis
HOSVD	Higher Order Singular Value Decomposition
ICA	Independent Component Analysis
LDA	Linear Discriminant Analysis
LS	Least Squares
MSE	Mean Squared Error
NMPCA	Non-negative Multilinear Principal Component Analysis
PCA	Principal Component Analysis
PSNR	Peak Signal-to-noise Ratio
RMPCA	Robust Multilinear Principal Component Analysis
SMT	Sequential Mode Truncation
SSS	Small Sample Size problem
TR1DA	Tensor Rank-one Discriminant Analysis
TROD	Tensor Rank-one Decomposition
TTP	Tensor-to-Tensor projection
TV	Total Variation
TVP	Tensor-to-Vector projection
UMLDA	Uncorrelated Multilinear Discriminant Analysis
UMPCA	Uncorrelated Multilinear Principal Component Analysis

REFERENCES

[1] H. Lu, K. N. Plataniotis and A. Venetsanopoulos, *Multilinear Subspace Learning*, Chapman and Hall/CRC, 2013.

[2] K. Pearson, "LIII. On lines and planes of closest fit to systems of points in space," vol. 2, p. 559-572, 1901.

[3] I. T. Jolliffe, *Principal Component Analysis*, Springer New York, 2010.

[4] amoeba (https://stats.stackexchange.com/users/28666/amoeba), *Making sense of principal component analysis, eigenvectors and eigenvalues.*

[5] R. A. F, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics,* vol. 7, p. 179–188, 1936.

[6] A. Hyvärinen and E. Oja, "Independent component analysis: algorithms and applications," *Neural Networks,* vol. 13, p. 411–430, 2000.

[7] H. Hotelling, "Relations Between Two Sets of Variates," *Biometrika,* vol. 28, p. 321, 1936.

[8] B. Schölkopf, A. Smola and K.-R. Müller, "Nonlinear Component Analysis as a Kernel Eigenvalue Problem," *Neural Computation,* vol. 10, p. 1299–1319, 1998.

[9] S. Mika, "Kernel fisher discriminants," 2003.

[10] F. R. Bach and M. I. Jordan, "Kernel independent component analysis," *Journal of machine learning research,* vol. 3, p. 1–48, 2002.

[11] P. L. Lai and C. Fyfe, "Kernel and nonlinear canonical correlation analysis," *International Journal of Neural Systems,* vol. 10, p. 365–377, 2000.

[12] M. Belkin and P. Niyogi, "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation," *Neural Computation,* vol. 15, p. 1373–1396, 2003.

[13] S. T. Roweis and L. K. Saul, "Nonlinear Dimensionality Reduction by Locally Linear Embedding," *Science,* vol. 290, p. 2323–2326, 2000.

[14] J. B. Tenenbaum, V. de Silva and J. C. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science,* vol. 290, p. 2319–2323, 2000.

[15] Y. Panagakis, J. Kossaifi, G. G. Chrysos, J. Oldfield, M. A. Nicolaou, A. Anandkumar and S. Zafeiriou, "Tensor Methods in Computer Vision and Deep Learning," *Proc. IEEE,* vol. 109, p. 863–890, 2021.

[16] H. Lu, K. N. Plataniotis and A. N. Venetsanopoulos, "Multilinear Principal Component Analysis of Tensor Objects for Recognition," *IEEE Transactions on Neural Networks*, vol. 19, p. 18-39, 2008.

[17] H. Lu, K. N. Plataniotis and A. N. Venetsanopoulos, "Uncorrelated Multilinear Principal Component Analysis for Unsupervised Multilinear Subspace Learning," *IEEE Transactions on Neural Networks,* vol. 20, p. 1820–1836, 2009.

[18] A. Shashua and A. Levin, "Linear image coding for regression and classification using the tensor-rank principle," *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* 2001.

[19] Y. Panagakis, C. Kotropoulos and G. R. Arce, "Non-Negative Multilinear Principal Component Analysis of Auditory Temporal Modulations for Music Genre Classification," *IEEE Transactions on Audio, Speech, and Language Processing,* vol. 18, p. 576–588, 2010.

[20] K. Inoue, K. Hara and K. Urahama, "Robust multilinear principal component analysis," *IEEE 12th International Conference on Computer Vision*, p. 591-597, 2009.

[21] Y. Pang, X. Li and Y. Yuan, "Robust Tensor Analysis With L1-Norm," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 20, p. 172–178, 2010.

[22] S. Yan, D. Xu, Q. Yang, L. Zhang, X. Tang and H.-J. Zhang, "Discriminant Analysis with Tensor Representation," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* 2005.

[23] D. Tao, X. Li, X. Wu and S. J. Maybank, "General Tensor Discriminant Analysis and Gabor Features for Gait Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 29, p. 1700–1715, 2007.

[24] H. Lu, K. N. Plataniotis and A. N. Venetsanopoulos, "Uncorrelated Multilinear Discriminant Analysis With Regularization and Aggregation for Tensor Object Recognition," *IEEE Transactions on Neural Networks,* vol. 20, p. 103–123, 2009.

[25] D. Tao, X. Li, X. Wu and S. Maybank, "Tensor Rank One Discriminant Analysis—A convergent

method for discriminative multilinear subspace selection," *Neurocomputing,* vol. 71, p. 1866–1882, 2008.

[26] H. Lu, "Learning Modewise Independent Components from Tensor Data Using Multilinear Mixing Model," *Advanced Information Systems Engineering*, p. 288–303, 2013.

[27] H. Wang, "Local Two-Dimensional Canonical Correlation Analysis," *IEEE Signal Processing Letters,* vol. 17, p. 921–924, 2010.

[28] L. Gang, Z. Yong, L. Yan-Lei and D. Jing, "Three Dimensional Canonical Correlation Analysis and Its Application to Facial Expression Recognition," *Communications in Computer and Information Science*, p. 56–61,2011.

[29] H. Lu, "Learning Canonical Correlations of Paired Tensor Sets via Tensor-to-Vector Projection," in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, Beijing, 2013.

[30] S. Yan, D. Xu, B. Zhang, H.-j. Zhang, Q. Yang and S. Lin, "Graph Embedding and Extensions: A General Framework for Dimensionality Reduction," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 29, p. 40–51, 2007.

[31] G. Hua, P. A. Viola and S. M. Drucker, "Face Recognition using Discriminatively Trained Orthogonal Rank One Tensor Projections," *2007 IEEE Conference on Computer Vision and Pattern Recognition*, p. 1–8, 2007.

[32] S. Wu, W. Li, Z. Wei and J. Yang, "Local Discriminative Orthogonal Rank-One Tensor Projection for image feature extraction," in *The First Asian Conference on Pattern Recognition*, p. 367-371 ,2011.

[33] D. Xu, S. Lin, S. Yan and X. Tang, "Rank-one Projections with Adaptive Margins for Face Recognition," *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics),* no. 5, p. 1226–1236, 2007.

[34] T. G. Kolda and B. W. Bader, "Tensor Decompositions and Applications," *SIAM Review,* vol. 51, p. 455–500, 2009.

[35] J. Håstad, "Tensor rank is NP-complete," *Journal of Algorithms,* vol. 11, p. 644–654, 1990.

[36] L. D. Lathauwer, B. D. Moor and J. Vandewalle, "A Multilinear Singular Value Decomposition," *SIAM Journal on Matrix Analysis and Applications,* vol. 21, p. 1253–1278, 2000.

[37] D. Luo, C. Ding and H. Huang, "Symmetric two dimensional linear discriminant analysis (2DLDA)," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, p. 2820–2827, 2009.

[38] F. D. la Torre, "A Least-Squares Framework for Component Analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 34, p. 1041–1055, 2012.

[39] Bathe, *Numerical methods in finite element analysis*, Englewood Cliffs, N.J: Prentice-Hall, 1976.

[40] K.-J. Bathe, "The subspace iteration method – Revisited," *Computers & Structures,* vol. 126, p. 177–183, 2013.

[41] W. Guo, I. Kotsia and I. Patras, "Tensor Learning for Regression," *IEEE Transactions on Image Processing,* vol. 21, p. 816–827, 2012.

[42] J. Kossaifi, Z. C. Lipton, A. Kolbeinsson, A. Khanna, T. Furlanello and A. Anandkumar, "Tensor regression networks," *Journal of Machine Learning Research (JMLR),* vol. 21, p. 21, 2020.

[43] J. Ye, "Least squares linear discriminant analysis," in *Proceedings of the 24th international conference on Machine learning - ICML*, p. 1087–1093, 2007.

[44] A. S. Georghiades, P. N. Belhumeur and D. J. Kriegman, "From few to many: illumination cone models for face recognition under variable lighting and pose," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 23, p. 643–660, 2001.

[45] K.-C. Lee, J. Ho and D. J. Kriegman, "Acquiring linear subspaces for face recognition under variable lighting," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 27, p. 684–698, 2005.

[46] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in *Proceedings of 1994 IEEE Workshop on Applications of Computer Vision*, p. 138–142, 1994.

[47] C. R. Harris, K. J. Millman, S. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke and T. E. Oliphant, "Array programming with NumPy," *Nat.,* vol. 585, p. 357–362, 2020.

[48] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner,

L. Fang, J. Bai and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," *Advances in neural information processing systems 32*, 2019.

[49] J. Kossaifi, Y. Panagakis, A. Anandkumar and M. Pantic, "TensorLy: Tensor Learning in Python," *J. Mach. Learn. Res.,* vol. 20, p. 26:1–26:6, 2019.

[50] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, A. Vijaykumar, A. P. Bardelli, A. Rothberg, A. Hilboll, A. Kloeckner, A. Scopatz, A. Lee, A. Rokem, C. N. Woods, C. Fulton, C. Masson, C. Häggström, C. Fitzgerald, D. A. Nicholson, D. R. Hagen, D. V. Pasechnik, E. Olivetti, E. Martin, E. Wieser, F. Silva, F. Lenders, F. Wilhelm, G. Young, G. A. Price, G.-L. Ingold, G. E. Allen, G. R. Lee, H. Audren, I. Probst, J. P. Dietrich, J. Silterra, J. T. Webber, J. Slavič, J. Nothman, J. Buchner, J. Kulick, J. L. Schönberger, J. V. de Miranda Cardoso, J. Reimer, J. Harrington, J. L. C. Rodríguez, J. Nunez-Iglesias, J. Kuczynski, K. Tritz, M. Thoma, M. Newville, M. Kümmerer, M. Bolingbroke, M. Tartre, M. Pak, N. J. Smith, N. Nowaczyk, N. Shebanov, O. Pavlyk, P. A. Brodtkorb, P. Lee, R. T. McGibbon, R. Feldbauer, S. Lewis, S. Tygier, S. Sievert, S. Vigna, S. Peterson, S. More, T. Pudlik, T. Oshima, T. J. Pingel, T. P. Robitaille, T. Spura, T. R. Jones, T. Cera, T. Leslie, T. Zito, T. Krauss, U. Upadhyay, Y. O. Halchenko and Y. Vázquez-Baeza, "SciPy 1.0: fundamental algorithms for scientific computing in Python," *Nature Methods,* vol. 17, p. 261–272, 2020.

[51] P. Umesh, "Image processing in python," *CSI Communications,* vol. 23, 2012.

[52] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," *Computing in Science & Engineering,* vol. 9, p. 90–95, 2007.

[53] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart and T. Yu, "scikit-image: image processing in Python," *PeerJ,* vol. 2, p. e453, 2014.

[54] J. Bush, "Bregman algorithms," *Senior Thesis. University of California, Santa Barbara,* 2011.

[55] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and É. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* vol. 12, p. 2825-2830, 2011.