Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Computer Science and Statistics

Modelling Large-scale Datasets Using Principal Component Analysis

Salaheddin Alakkari

May 2020

A dissertation submitted in partial fulfilment
of the requirements for the degree of
Doctor of Philosophy

Declaration

Signed: Salaheddin Date: 13/5/2020

Abstract

Principal Component Analysis (PCA) is one of the most well-known unsupervised learning techniques used for dimensionality reduction and feature extraction. The main task of PCA is to compute a low-dimensional space that captures the maximal variability in the input dataset. Such a holistic linear representation is optimal in terms of the mean-squared-error. The basis vectors that form such a space correspond to the k most significant eigenvectors of the sample covariance matrix. However, computing such eigenvectors is computationally expensive with quadratic computational dependence on the data size. The ever-increasing size of datasets necessitates investigating reduced-complexity methods to find such eigenvectors.

A common treatment is to apply streaming PCA methods which aim to approximate the eigenvectors based on a single data pass with a linear computational cost. However, state-of-the-art streaming approaches are highly sequential and assume that samples are independent and identically distributed. In the first part of this thesis, we investigate the convergence of such methods when extended to the mini-batch mode, which is superior to the traditional fully online mode in terms of computation run-time. Furthermore, we propose an acceleration scheme for mini-batch streaming methods that are based on the Stochastic Gradient Approximation (SGA). Such methods provide the cheapest computational cost compared to other streaming algorithms. Based on empirical evaluation using the spiked covariance model and benchmark datasets, we show that applying our scheme significantly enhances the convergence of the original techniques in addition to outperforming other state-of-the-art methods.

In the second part, we investigate the performance of PCA when applied in a partitioning manner in which attributes are divided into a number of subsets, and then the standard approach is performed on each subset separately. We study two strategies for mapping attributes to different sets, namely, Cell-based PCA (CPCA) where samples are spatially divided into smaller blocks and Band-based PCA (BPCA) where attributes are partitioned based on their values distribution. We show that such models have several advantages over the holistic approach, including enhanced reconstruction quality and increased scalability. We also find that the baseline model, obtained when randomly mapping attributes, is analogous to the holistic PCA which entails a more practical and parallel alternative to streaming PCA paradigms. Not only are these methods beneficial for data compression but they also provide lightweight representations that would enhance the accuracy and training time of deep learning models.

Time-varying datasets of various physical observations are also addressed. We theoretically draw the analogy between many analytic physical models and the PCA eigenvalue problem. It is shown that, for a wide range of physical phenomena, the eigenvectors derived using PCA are analogous to the analytic physical model. Since time-varying datasets are no exception from the curse of dimensionality, we further evaluate the performance of streaming PCA methods on many time-varying physical observations.

Acknowledgements

I am infinitely grateful to my supervisor Dr. John Dingliana for all his invaluable advice and guidance. This work would not have been possible without him.

I also want to express my deepest gratitude to my parents for their support and help during my research. They have always believed in me and taught me that the real strength lies in our knowledge. Special thanks to my wife, who has been extremely supportive of me throughout this entire process and has made countless sacrifices to help me get to this point.

Contents

List of Figures

List of Tables

Nomenclature

X	A matrix of n columns and d rows representing a dataset of n samples where each sample has d features (attributes) represented as a column vector.
C	A square matrix of size $d \times d$ representing the covariance matrix of the dataset X.
V	A matrix of k columns and d rows where the ith column corresponds to the ith most significant eigenvector of the covariance matrix C computed using standard PCA.
W	A matrix of k columns and d rows where the ith column corresponds to the ith most significant eigenvector of the covariance matrix C estimated using a complexity-optimized PCA method.
x	A sample of dataset X represented as a column vector of d features.
$\bar{x}$	The sample mean of dataset X.
v	The most significant eigenvector of the covariance matrix C computed using standard PCA.
w	The most significant eigenvector of the covariance matrix C computed using a complexity-optimized PCA method.
n	The number of samples in the dataset X.
d	The number of features (attributes) in the dataset X.
k	The number of significant eigenvectors to be computed using standard PCA or a complexity-optimized PCA method.
λ_i	The eigenvalue of the ith significant eigenvector.
Δ_i	The eigengap between eigenvectors i and $i+1$ defined as $\Delta_i = \lambda_i - \lambda_{i+1}$.
η_t	The learning rate scheduler of the streaming PCA method.
α_t	The learning rate scheduler of our acceleration scheme.

1 Introduction

1.1 Overview

Principal Component Analysis (PCA), invented by Karl Pearson in 1901 [Pea01], is one of the oldest and most well-known machine learning methods. It has found its way into a plethora of applications in almost all scientific fields. The task of PCA is to automatically find a linear representation that describes most variations in the input dataset using a minimal number of orthogonal basis vectors. The main difference that distinguishes PCA from other basis function representations, such as Fourier and Discrete Cosine transforms is the fact that the basis derived using PCA is data-dependent whereas the other transforms consider a global basis for representing any input data (which consists mainly of sinusoidal functions of different frequencies). This data-dependence is to ensure a dimensionality reduction scheme with minimal reconstruction errors [Jol02]. However, the role of PCA lies not only on dimensionality reduction and data representation but also on feature extraction. In other words, the PCA basis vectors are not merely dimensions, but they also correspond to the key features in the data samples. Such features are of considerable significance to many pattern recognition tasks. Furthermore, the mathematical interpretation of Hebbian law, one of the fundamental theories in Psychobiology, demonstrates that such features are directly related to the human perception and is still within the central focus of many neuroscientific studies.

Acquiring such basis vectors is typically achieved by decomposing the covariance matrix into its significant eigenvectors. The covariance matrix is a square matrix consisting of covariances between attributes in the dataset. This implies that the computation of the eigenvectors has a quadratic complexity dependence on the data size. Considering a dataset of n samples and d attributes, the standard approach to PCA requires $\mathcal{O}\left(nd \min\left(n, d\right)\right)$ floating-point operations and $\mathcal{O}\left(nd + \min\left(n, d\right)^2\right)$ memory space. If n and d are small enough (within the order of hundreds or thousands), then the computation is feasible, and the problem no longer holds. However, modern datasets are becoming so massive in size that even fitting a few samples into a typical modern machine space may not be possible. Moreover, many datasets are of a streaming nature which means *n* may scale to infinity. Many algorithms have been developed to find the optimal eigenvectors with lower memory space and time complexity. We can classify each of these algorithms into two main categories: Offline and online algorithms. Offline techniques compute the optimal eigenvectors using a number of epochs where at each epoch a single pass over the entire dataset is performed. Since such methods require the presence of all samples, they provide a suitable solution when the dataset is manageable within the machine memory space. Online approaches (also referred to as memory-limited or streaming approaches), on the other hand, aim to provide an acceptable convergence after only a single data pass. Thus, they are more appropriate when dealing with streaming data or when data samples are too large to fit into the memory space. While a streaming approach provides potential solutions to a vast number of practical use cases, existing online algorithms cannot be applied efficiently in many of these scenarios for many reasons. One reason is the fact that state-of-the-art streaming PCA methods are highly sequential. Many real-time scientific simulations require in situ analysis of the input time-steps, which may involve interactive visualization of the significant eigenvectors up to the recent time-stamp. Current solutions are far from such a target performance, particularly when sample dimensionality is too large, which is the case in most physical simulations. Besides, many streaming PCA algorithms involve a learning rate hyperparameter that needs to be tuned. Tuning such a hyperparameter may be challenging, particularly when considering large-scale time-varying datasets.

The role of PCA in modelling physical time-varying phenomena is another interesting area that is still under research. From a physical point of view, eigenvalue problems have been widely used for modelling an endless amount of real-world applications and physical phenomena. There is no better evidence of their importance than the fact that 20 Nobel prizes went to physicists whose significant achievements are related to eigenvalue problems [Tre11]. In fact, many landmark models in Physics are actually eigenvalue problems such as Dirac and Schrödinger equations which describe particle motion and properties over time. Before the computing era, obtaining the eigenfunctions that govern a physical model was done analytically. In many cases, acquiring the analytic solution to the problem at hand might be challenging. Since finding the eigenvectors of observations using PCA also involves solving an eigenvalue problem, an intuitive question would be: Are the eigenvectors derived from observations of a physical phenomenon using PCA related to the underlying physical model? Can we automatically find the analytic physical model using PCA? Literature reveals very interesting findings in that regard. Cohen and Moerner [CM07] showed that when applying PCA to video frames of single-molecule fluctuations, the obtained eigenvectors were very similar to the analytic solution. However, such time-varying observations may demand a large number of time-steps to cover the appropriate time-scale and sampling frequencies of the different eigenfunctions, not to mention the high dimensionality problem. Hence, computation barriers entail streaming PCA as a more practical approach in such scenarios. To the best of my knowledge, experimental studies regarding the reliability of streaming PCA for time-varying datasets are not widely explored in the literature.

From another perspective, PCA is considered to be a holistic representation. The implication of such holism is that in practice, the number of eigenvectors that well-represent the data samples may be relatively large and subjective to determine. Such a large number of eigenvectors entails not only higher reconstruction costs but also significant overhead computation for streaming PCA. In fact, many streaming PCA methods have a quadratic computational dependence on the number of eigenvectors due to the orthogonalization processes involved. In addition, these methods are tested for only computing a few eigenvectors. What will be the implications when applying PCA in a partitioning mode? Many representation schemes were found to be more efficient when applied in a partitioned manner, such as the blockwise DCT used in the JPEG compression standard. This has several key advantages over the traditional holistic scheme in terms of reduced computational cost due to its parallel nature and enhanced reconstruction quality. Surprisingly, PCA performance in the partitioning mode is not widely reported. In addition, such a mode of computation raises two main questions. First of all, on what basis are attributes mapped to different partitions? The second question concerns the relation between the partitioned eigenvectors and the holistic ones. Can we approximate the holistic eigenvectors using the partitioning scheme? Can such a mode of computation be the perfect parallel alternative to streaming PCA approaches?

This thesis investigates possible solutions and answers to the above-mentioned problems. We first discuss the mini-batch mode for reducing the run-time of streaming PCA and whether such a mode can lead to improvements in convergence rates. We further propose an acceleration scheme for a wide range of streaming PCA methods that are based on the Stochastic Gradient Approximation (SGA). We then discuss the performance of PCA when applied in a partitioning manner and its implication when used as a data representation scheme for deep learning models. Time-varying systems are also addressed from two perspectives. Firstly, we study the relation between PCA eigenvectors derived directly from time-varying observations and the underlying physical model and to what extent can such eigenvectors reflect the analytic model. Secondly, we examine the performance of state-of-the-art streaming PCA methods when applied on time-varying observations in order to estimate the significant eigenvectors up to the recent time-step efficiently. In addition, a performance evaluation of the partitioning approach in such cases is also provided. Throughout the thesis, we employ empirical evaluation strategies using benchmark datasets and other generative models such as the spiked covariance model in order to assess performance. Whenever theoretical analyses are conducted, we carefully show their consistency with the literature.

1.2 Scope

The main focus of this thesis is on exploring effective and efficient strategies for utilizing PCA as a computationally-economical unsupervised machine learning and dimensionality reduction paradigm for deriving meaningful lightweight representations of large scale datasets. Despite the fact that many reduced-complexity PCA methods have been proposed in the literature, current state-of-the-art solutions suffer many limitations related to their scalability and runtime. My thesis hence explores more practical and scalable solutions. The main research gaps that my thesis addresses are summarized below:

1. Current state-of-the-art streaming PCA methods are highly sequential and do not scale well when increasing the dimensionality and number of eigenvectors. Efficient parallel solutions are not widely explored.

2. Most streaming PCA approaches are parametric and hence suffer from the parameter-tuning problem. In many cases, finding the optimal settings for such parameters requires a preprocessing data pass which violates the streaming learning main condition.

3. There is a considerable gap between the theoretical analysis of these approaches and the empirical evaluation. Many studies do not even provide any empirical evaluation and focus mainly on the theoretical convergence guarantee. In many cases, the theoretical analysis is not consistent with the empirical evaluation.

4. State-of-the-art streaming PCA methods assume that samples are independent and identically distributed. The performance of such approaches is not evaluated on time-varying data which is a core application for which such techniques are facilitated.

5. PCA is applied as a holistic representation scheme. Such a holism may imply a large number of eigenvectors in order to reach satisfactory reconstruction quality. Research studies lack reports on the use of PCA in a partitioning manner.

6. State-of-the-art deep learning models such as Convolutional Neural Networks (CNN) indirectly perform dimensionality reduction for feature extraction. However, the parametric and stochastic nature of such models makes it hard to arrive at an optimal (inner) representation. From another perspective, the role of deterministic dimensionality reduction schemes in deriving meaningful and simple representations that would enhance the perception of such models has not been widely examined.

1.3 Research Question

What strategies can be used for modelling large-scale datasets effectively and efficiently using PCA? Particularly, my thesis explores answers to the following more specific questions:

1. What techniques can be used in order to enhance the convergence rates and reduce the run-time of streaming PCA methods? How can we evaluate such techniques?
2. What advantages can be gained when employing the partitioning-based PCA approach over the conventional, holistic scheme? Can a full parallel solution to the streaming PCA problem lie behind such an approach?
3. How can deterministic dimensionality reduction approaches, such as PCA, lead to better accuracy in deep learning classifiers?
4. What is the relationship between the eigenvectors derived from observations of a physical phenomenon using PCA and the actual physical model? To what extent can the PCA solution reflect the analytic model?
5. How reliable are streaming PCA approaches in terms of convergence when considering time-varying observations? Can we get an acceptable estimation of the significant k eigenvectors up to the recent time-step?

1.4 Contributions

The main contributions of this work are listed below:

1. We propose an acceleration scheme for a wide range of streaming PCA methods that can achieve a significantly better convergence compared to state-of-the-art streaming PCA methods.
2. We study the parallel setting of PCA and show the efficacy of such an approach for modelling and compressing large-scale datasets.
3. We show that the parallel PCA model can significantly enhance the accuracy of deep learning classifiers, particularly when considering limited training examples.
4. We develop a novel deep learning classifier that can be efficiently trained with limited reduced-size examples and show how such a classifier outperforms state-of-the-art deep learning classifiers when considering limited training examples.
5. For time-varying physical systems, we theoretically draw the analogy between PCA and well-known physical models that governs a wide range of phenomena.
6. We provide an empirical evaluation of streaming PCA methods for such a type of datasets.

1.5 Peer Reviewed Publications

A list of peer-reviewed publications that emanated from this thesis are provided below:

[AD16a] Salaheddin Alakkari and John Dingliana. "Volume Rendering Using Principal Component Analysis." In: *EuroVis (Posters)*. 2016, pp. 85–87.

[AD16b] Salaheddin Alakkari and John Dingliana. "Volume visualization using principal component analysis". In: *Proceedings of the Eurographics Workshop on Visual Computing for Biology and Medicine*. Eurographics Association. 2016, pp. 53–57.

[AD18a] Salaheddin Alakkari and John Dingliana. "A Multi-View Image-Based Volume Visualization Technique". In: *IEEEVis (Posters)*. 2018.

[AD18b] Salaheddin Alakkari and John Dingliana. "An accelerated online PCA with O (1) complexity for learning molecular dynamics data". In: *Proceedings of the Workshop on Molecular Graphics and Visual Analysis of Molecular Data*. Eurographics Association. 2018, pp. 1–8.

[AD18c] Salaheddin Alakkari and John Dingliana. "Principal Component Analysis Techniques for Visualization of Volumetric Data". In: *Advances in Principal Component Analysis*. Springer, 2018, pp. 99–120.

[AD19a] Salaheddin Alakkari and John Dingliana. "An Acceleration Scheme for Mini-batch, Streaming PCA". In: *2019 British Machine Vision Conference (BMVC)*. British Machine Vision Association. 2019.

[AD19b] Salaheddin Alakkari and John Dingliana. "Modelling Large Scale Datasets Using Partiotioning-based PCA". In: *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2019.

1.6 Thesis Structure

The overall thesis is structured as follows:

Chapter 1 provides an introduction to the thesis with the main motivation behind this work and lists current research gaps, main research questions, and peer-reviewed publications that emanated from this work.

Chapter 2 gives a structured literature review with a detailed background on the topic and visits other related approaches from different disciplines, including neural networks, data compression and Physics.

Chapter 3 describes an acceleration scheme for enhancing the convergence of streaming PCA methods that are based on the Stochastic Gradient Approximation (SGA). Mini-batching is considered for increasing the speed-up. It also provides an empirical evaluation strategy for streaming PCA and studies the convergence of several state-of-the-art approaches when extended to work in the mini-batch mode.

Chapter 4 studies the performance of PCA when applied in a partitioning manner. Attributes are divided into different sets, and then the standard PCA is performed on each set separately. Extensive experimental studies are conducted on two partitioning strategies: *Cell-based PCA (CPCA)* which spatially divides the images into smaller blocks and *Band-based PCA (BPCA)* which partitions the samples based on the attributes values distribution. It also considers the baseline performance when attributes are divided in a random manner.

Chapter 5 investigates the use of the cell-based model as a lightweight representation for deep learning classifiers, specifically in the area of face recognition. A deep convolutional neural network classifier, referred to as RedNet, is engineered particularly for learning with limited reduced-size examples. The performance of this classifier is examined when representing face images using CPCA and the traditional image downsampling representation. We compare the accuracy of RedNet with the accuracy of state-of-the-art deep CNNs trained from scratch using high-resolution images.

Chapter 6 covers modelling large-scale time-varying physical systems. An investigation of the analogy between PCA and landmark physical models is provided. After that, the efficacy of both state-of-the-art streaming PCA methods and the partitioning approach for modelling time-varying observations is further discussed.

Chapter 7 presents the overall conclusions and summarizes future research goals.

2 Background

This chapter gives a detailed background with a structured literature review of the topic. It also visits other related approaches from different disciplines that will be used in developing our techniques. It is worth mentioning that each technical chapter in this thesis will provide further related work in more specific areas.

2.1 The standard Approach to PCA

The standard approach to PCA is as follows. Given data samples $X = [x_1\, x_2 \cdots x_n] \in \mathbb{R}^{d\times n}$, where each sample is represented in a column vector format, the covariance matrix is defined as

$$C = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^T, \tag{2.1}$$

where $\bar{x}$ is the sample mean. We will assume in the sequel of this thesis that all samples are centered and hence there is no need to subtract the sample mean explicitly. After computing the covariance matrix, we can find the optimal set of basis vectors, referred to as eigenvectors, that covers the maximal variability in the input samples by extracting the significant eigenvectors of the covariance matrix C. Eigenvectors are extracted by solving the following eigenvalue equation

$$(C - \lambda I)\, v = 0;\ v^T v = 1, \tag{2.2}$$

where $v \in \mathbb{R}^d$ is the eigenvector and λ is its corresponding eigenvalue. Eigenvalues describe the variance maintained by the corresponding eigenvectors. Hence, we are interested in the subset of k eigenvectors that have the highest eigenvalues $V = [v_1\, v_2 \cdots v_k];\ k \ll n$. After that, any sample $x \in \mathbb{R}^d$ of X can be encoded using its k-dimensional projection values (referred to as *scores*) as follows

$$P = V^T(x - \bar{x}). \tag{2.3}$$

We can then reconstruct the sample as follows

$$x_{reconstructed} = VP + \bar{x}. \tag{2.4}$$

Such a reconstruction gives the minimal reconstruction errors using a space $V \in \mathbb{R}^{d \times k}$ of k basis vectors [Jol02].

Duality in PCA

Since in the case of $n \ll d$, C will be of rank $n-1$ (considering that all samples are centered). Hence, there are only $n-1$ meaningful eigenvectors that can be extracted from Eq. (2.2) and since C is of size $d \times d$, solving Eq. (2.2) becomes computationally expensive. We can find such eigenvectors from the dual eigenspace by computing the $n \times n$ matrix $X^T X$ and then solving the eigenvalue problem

$$\left(X^T X - (n-1)\lambda I\right) v_{dual} = 0 \tag{2.5}$$

$$\Rightarrow X^T X v_{dual} = (n-1)\lambda v_{dual}; \; v_{dual}^T v_{dual} = 1. \tag{2.6}$$

After extracting the dual eigenvectors, one can note that by multiplying each side of Eq. (2.6) by X, we have

$$\begin{aligned} XX^T X v_{dual} &= (n-1)\lambda X v_{dual} \\ \Rightarrow \frac{1}{n-1} XX^T \left(X v_{dual}\right) &= \lambda \left(X v_{dual}\right) \\ \Rightarrow C \left(X v_{dual}\right) &= \lambda \left(X v_{dual}\right) \\ \Rightarrow \left(C - \lambda I\right)\left(X v_{dual}\right) &= 0 \end{aligned}$$

which implies that

$$v = X v_{dual}. \tag{2.7}$$

Thus, when $n \ll d$, we only need to extract the dual eigenvectors using Eq. (2.6) and then compute the real eigenvectors using Eq. (2.7). Only the first few eigenvectors $V_k = [v_1 \; v_2 \ldots v_k]$, $k \ll n \ll d$ are chosen to represent the eigenspace, those with larger eigenvalues.

2.1.1 Solving the PCA Eigenvalue Problem

Solving equations 2.2 and 2.6 can be achieved by applying Eigenvalue Decomposition (EVD) on the covariance matrix (or dual covariance matrix) or performing Singular Value Decomposition (SVD) directly on the data samples without the need of computing the covariance matrix. Both techniques have a computational complexity of $\mathcal{O}(nd \min(n, d))$. In terms of memory space, SVD requires $\mathcal{O}(nd)$ memory usage and EVD requires extra $\mathcal{O}(\min(n, d)^2)$ space for storing the covariance matrix. Although such standard approaches give the best quality performance in terms of optimizing the objective function, handling dataset of very large numbers of samples n and dimensionality d becomes computationally infeasible due to the quadratic complexity dependence on the data size. In addition, many datasets are of streaming nature and hence cannot be processed in the batch mode. Most datasets nowadays suffer at least one of these conditions.

Due to the high computational cost of SVD and EVD, many reduced complexity PCA algorithms were proposed. We can categories each of these algorithms into two main groups: Offline and online algorithms. Offline techniques compute the optimal eigenvector using a number of epochs wherein each epoch a single data pass is performed. The power method is one of the most popular and simplest offline techniques for finding the most significant eigenvector of the covariance matrix [GV12]. The main downside of this algorithm is that in many cases, it may converge very slowly. Shamir proposed a refined PCA algorithm that was proven to converge faster than the power method [Sha15]. Online approaches (also referred to as memory-limited or streaming approaches), on the other hand, aim to provide an acceptable convergence after only a single data pass. Thus, they are more appropriate when dealing with streaming data or when data samples are too large to fit into the memory space. In the following sections, we will provide a more detailed review on state-of-the-art complexity-reduced PCA approaches.

2.1.2 A Note on Lower Eigenvectors

One important point to highlight is that most studies focus mainly on the most significant eigenvectors with a little attention paid to the least significant ones. In fact, finding such eigenvectors was shown to play a key role in detecting outliers and non-belonging samples since they are perpendicular to the best fitting hyperplane. Jollife [Jol02] pointed out in his book that the principal components corresponding to the smallest eigenvalues (variances) are not "unstructured left -overs" after extracting the higher eigenvectors and that they can be useful in detecting outliers. The first report on the use of lowest eigenvectors in the literature is credited to Gnanadesikan and Wilk 1969 [GW69]. Based on this work, Gnanadesikan [GK72] stated that "with p-dimensional data, the projection onto the smallest principal component would be relevant for studying the deviation of an observation from a hyperplane of closest fit". More recently, Izenman and Shen used the smallest kernel principal components for outlier detection as a generalization of the linear case [IS09]. Alakkari et al. found that the least significant eigenface can be used as a basis for discriminating between face and non-face images [AGC15].

2.2 Offline Complexity-optimized PCA Approaches

The main idea of offline methods is to compute dominant eigenvectors using a small number of iterations where at each iteration, one or more data pass is required. Such approaches provide very good approximation and enjoy solid theoretical ground and convergence guarantee. However, there is one main disadvantage concerning the burn-in phase problem, which is a state where the convergence rate becomes extremely slow even after performing multiple iterations. Such a problem usually occurs at the beginning of the learning process or when the difference in eigenvalues between eigenvectors is considerably small. Since at each iteration, a full data pass is performed, this may significantly increase the computational cost, specifically in cases of very high dimensional data. Hence, such approaches are impractical when considering large-scale datasets.

2.2.1 The Power Method

The power method, also known as the power iteration, is probably the most well-known offline complexity-optimized PCA approaches. It was coined by the Austrian mathematician Richard von Mises in 1929 [MP29]. It is widely used as a benchmark model for evaluating state-of-the-art complexity-optimized PCA methods. The basic idea of the power method is to optimize the following objective function

$$\max_{w^T w=1} H(w) = w^T C w. \tag{2.8}$$

By directly applying the gradient ascent, one gets the following update rule

$$\begin{aligned} w_{t+1} &= w_t + \frac{\eta}{2}\frac{\partial}{\partial w_t} H(w_t) \\ w_{t+1} &= w_t + \frac{\eta}{2}\frac{\partial}{\partial w_t}\left(w_t^T C w_t\right) \\ &= w_t + \eta C w_t \end{aligned} \tag{2.9}$$

Setting the learning rate η to infinity, we arrive to the power method update rule

$$w_{t+1} = \frac{C w_t}{\|C w_t\|}. \tag{2.10}$$

Assuming that the initial eigenvector is a weighted sum of the actual eigenvectors, we have

$$w_0 = c_1 v_1 + c_2 v_2 + ... + c_n v_n. \tag{2.11}$$

One can note that

$$\begin{aligned} C w_t = C^t w_0 &= c_1 C^t v_1 + c_2 C^t v_2 + ... + c_n C^t v_n \\ &= c_1 \lambda_1^t v_1 + c_2 \lambda_2^t v_2 + ... + c_n \lambda_n^t v_n \\ &= c_1 \lambda_1^t \left(v_1 + \frac{c_2}{c_1}\left(\frac{\lambda_2}{\lambda_1}\right)^t v_2 + ... + \frac{c_n}{c_1}\left(\frac{\lambda_n}{\lambda_1}\right)^t v_n \right). \end{aligned} \tag{2.12}$$

Since $\lambda_k \leq \lambda_{k+1}$, the coefficients of the lower eigenvectors will converge to zero and hence Cw_t will converge to the most significant eigenvector v_1. The speed of convergence of this method is governed by the difference in eigenvalues between the first and second eigenvectors which is known as the eigengap $\Delta = \lambda_1 - \lambda_2$. Let T denote the number of iterations required to converge to an eigenspace with a precision of δ at most. From eq. 2.12, such a precision is dominated by the factor $\frac{\lambda_2}{\lambda_1}$ which can be expressed as follows

$$\delta > \left(\frac{\lambda_2}{\lambda_1}\right)^T \Rightarrow T > \frac{\log(\frac{1}{\delta})}{\log(\lambda_1) - \log(\lambda_2)} > \frac{\log(\frac{1}{\delta})}{\lambda_1 - \lambda_2} = \frac{\log(\frac{1}{\delta})}{\Delta},$$

from which we can conclude that the computational cost for arriving at δ precision is $\mathcal{O}(nd\frac{\log(\frac{1}{\delta})}{\Delta})$ since each iteration involves a single data pass of nd FLOPs. Hence the smaller the eigengap the slower the convergence rate of this technique.

2.2.2 Variance-reduced PCA

As discussed in the previous section, one of the main limitations of the power method is the fact that it converges very slowly when the eigengap between the first two dominant eigenvectors is relatively small. Shamir addressed this problem in [Sha15] and found that using variance-reduced gradient descent, proposed by Johnson and Zhang in [JZ13], in conjunction with the power method significantly enhanced the convergence rates even in cases of very small eigengaps. Consider the following stochastic update rule

$$w_{t+1} = \left(I + \eta x_{i_t} x_{i_t}^T\right) w_t, \tag{2.13}$$

where η is the learning rate and x_{i_t} is a randomly chosen sample at time t. This update rule is known as the Hebbian rule which will be discussed in more detail in Section 2.4.1. One can rewrite this equation as follows

$$w_{t+1} = (I + \eta C) w_t + \eta \left(x_{i_t} x_{i_t}^T - C\right) w_t \tag{2.14}$$

where C is the covariance matrix. The main idea of the variance-reduced PCA is to accelerate the convergence rate by rapidly decaying the second term in the right-hand-side as follows

$$w_{t+1} = \frac{(I + \eta C) w_t + \eta \left(x_{i_t} x_{i_t}^T - C\right)(w_t - \tilde{w}_s)}{\left\|(I + \eta C) w_t + \eta \left(x_{i_t} x_{i_t}^T - C\right)(w_t - \tilde{w}_s)\right\|}. \tag{2.15}$$

The algorithm performs a number of epochs where at each epoch a single power iteration step is performed on $\tilde{w}_s$ followed by m update operations as in eq. 2.15. The full pseudo code of the algorithm is provided in Algorithm 2.1. Theoretical analyses and empirical results showed that VR-PCA is less sensitive to the eigengap compared to the power method and hence is more practical. While this approach is parametric, a suitable η value can be easily obtained as suggested in [Sha15].

Algorithm 2.1: VR-PCA

```
Input: Data samples $X = [x_1\, x_2 \cdots x_n] \in \mathbb{R}^{d \times n}$, Learning rate $\eta$ and number of
       iterations $m$ per epoch
Output: The most significant eigenvector of $X$
1  Initialize $\tilde{w}_1$ from the unit sphere in $\mathbb{R}^d$;
2  for $s = 1, 2, \ldots$ do
3      $\tilde{u} = \left(\frac{1}{n} \sum_{i=1}^{n} x_i x_i^T\right) \tilde{w}_s$;
4      $w_1 = \tilde{w}_s$;
5      for $t = 1, 2, \ldots, m$ do
6          Pick a random sample $x_i$ form the dataset;
7          $w_{t+1} = \frac{w_t + \eta\left(x_i x_i^T (w_t - \tilde{w}_s) + \tilde{u}\right)}{\left\| w_t + \eta\left(x_i x_i^T (w_t - \tilde{w}_s) + \tilde{u}\right)\right\|}$;
8      end
9      $\tilde{w}_{s+1} = w_{m+1}$;
10 end
```

2.3 Streaming PCA Methods

Unlike offline methods, streaming PCA methods aim to approximate the most significant eigenvectors based on a single data pass which is the main problem under investigation in this thesis. In this section, we give a review of state-of-the-art streaming PCA approaches. These methods will be used for empirical evaluations later in this thesis.

2.3.1 Stochastic Gradient Approximation

The Stochastic Gradient Approximation (SGA) is one of the oldest attempts to address the online learning behaviour of PCA. Such approaches are based on the assumption that the covariance matrix can be approximated using an arbitrary sample x from the input dataset $\mathbb{E}\left(xx^T\right) = C$. The most well-known SGA techniques are the ones proposed by Krasulina [Kra69] and Oja [Oja82; OJE83]. Such methods are based on the normalized Hebbian rule which is defined as follows

$$w_{t+1} = \frac{w_t + \eta_t x_t x_t^T w_t}{\left\| w_t + \eta_t x_t x_t^T w_t \right\|}, \tag{2.16}$$

where the learning rate should obey Robbins-Monro conditions ($\sum_t^\infty \eta_t = \infty$ and $\sum_t^\infty \eta_t^2 < \infty$) [RM85]. One important point to mention is that in the literature, this update rule is mistakenly referred to as Oja's rule. We will discuss this point later in Section 2.4.1. A typical form of the learning rate is $\eta_t = c/t$ for some positive constant c. One main limitation of these techniques is their performance sensitivity to different choices of the learning rate, which may significantly affect the convergence rates. According to Balsubramani et al. [BDF13], the optimal convergence rates of this update rule is achieved when setting $c = \frac{a}{\Delta}$ where $a > 1$ and $\Delta = \lambda_1 - \lambda_2$ is the eigengap. This is consistent with the theoretical analysis in Plumbley's paper, where convergence was investigated in terms of Lyaponov functions [Plu95]. Such findings suggest that achieving the optimal learning rate requires a pre-processing data pass for computing the eigengap, which violates the streaming PCA main condition. Furthermore, each individual eigenvector (when computing multiple $k > 1$ eigenvectors) should have its own learning rate adding an extra hurdle.

The computational cost of these algorithms is $\mathcal{O}(kd)$ FLOP per update, where k is the number of eigenvectors to compute and d is the dimensionality. These techniques are considered the cheapest in terms of computational cost and space complexity. Generalizing such techniques to compute multiple eigenvectors ($k > 1$) can be achieved by replacing the normalization process by an orthonormalization process as follows

$$W_{t+1} = \text{Orthonormalize}\left(W_t + \eta_t x_t x_t^T W_t\right) \in \mathbb{R}^{d\times k}$$

where the orthonormalization is done either using Gram-Schmidt orthogonalization which requires $\mathcal{O}(k^2d)$ extra FLOPs or using deflation which costs $\mathcal{O}(kd)$ FLOPs but may suffer from round-off errors. It can be proven that the orthogonalization step is independent of convergence rate [Aro+12]. Hence, one does not need to apply the Gram-Schmidt process at each update, but rather we can perform this after every B number of updates. Other approaches that are based on SGA include the mini-batch (or block) power method, proposed by Mitliagkas et al. [MCJ13], which applies the power iteration on a mini-batch of the data that is visited only once and will never be revisited again. The main drawback of this method is that it requires relatively large mini-batch sizes B in order to achieve satisfactory convergence rates. They proved the convergence of this method when $B \geq n/\log(d)$. Li et al. [LLL16] showed that the convergence of mini-batch power might be significantly improved when using dynamic mini-batch sizes. De Sa et al. studied the convergence of state-of-the-art SGA methods after applying the momentum term for accelerating the SGA learning process [Xu+18]. They showed that using momentum term with the mini-batch power and reduced-variance power method (based on VR-PCA [Sha15]) produced better convergence rates over the original approaches. However, finding the optimal momentum factor requires computing the eigengap violating the online learning condition because of the extra pre-processing data pass.

2.3.2 Candid Covariance-free Incremental PCA

Candid Covariance-free Incremental PCA (CCIPCA), proposed by Weng et al. [WZH03], is an online PCA algorithm that is similar to Oja's method but is rather more efficient in that it is a parameter-free approach. The main idea of this method is to average the eigenvalue equation (2.2) over all time-steps as follows

$$w_{t+1} = \frac{1}{t} \sum_{i=1}^{t} x_i {x_i}^T \frac{w_i}{\|w_i\|}, \tag{2.17}$$

where $\|w_t\|$ corresponds to the eigenvalue estimation up to time-step t. This can be rewritten in a more compact recursive form as follows

$$w_{t+1} = \frac{t}{t+1} w_t + \frac{1}{t+1} x_t {x_t}^T \frac{w_t}{\|w_t\|}. \tag{2.18}$$

Generalizing the approach to update multiple eigenvectors is done using deflation where after updating the jth significant eigenvector, the current time-step x_t is replaced by $x_t - \sum_{k=1}^{j} W_t^k (W_t^k)^T x_t$. In case of non-stationary samples, an amnesic factor $\ell \geq 0$ is applied as follows

$$w_{t+1} = \frac{t-\ell}{t+1} w_t + \frac{1+\ell}{t+1} x_t {x_t}^T \frac{w_t}{\|w_t\|}, \tag{2.19}$$

where larger ℓ values give more importance to most recent samples. A typical range for ℓ in the case of non-stationary data distribution is between 2 and 4. While extending this method to work in mini-batch mode is possible, it has not been investigated in the literature.

2.3.3 Incremental PCA

Arora et al. [Aro+12] proposed a reduced-order Incremental PCA model. The main idea of their technique is to update the eigenvalues of the first k eigenvectors and then perform SVD on a reduced-order $(k+1) \times (k+1)$ matrix. Let S_t be the $k \times k$ diagonal matrix with the eigenvalues estimation (up to time-step t) in its diagonal elements, IPCA computes the following matrix

$$Q_{t+1} = \begin{bmatrix} S_t + \tilde{x}\tilde{x}^T & \|r\|\,\tilde{x} \\ \|r\|\,\tilde{x}^T & \|r\|^2 \end{bmatrix}, \tag{2.20}$$

where $\tilde{x} = W_t^T x_t$ contains the projection values of the recent time-step and $r = x_t - W_t W_t^T x_t$ is the residual vector. After that, SVD is performed on the matrix Q_{t+1} and outputs S_{t+1} which contains $k+1$ eigenvalues and U_{t+1} which contains $k+1$ eigenvectors of dimensionality $k+1$. The eigenvectors are then updated as follows

$$W_{t+1} = \begin{bmatrix} W_t & \frac{r}{\|r\|} \end{bmatrix} U_{t+1}. \tag{2.21}$$

Since W_{t+1} will contain $k+1$ eigenvectors, the final eigenvectors are sorted based on the eigenvalues in S_{t+1} and the eigenvector associated with the lowest eigenvalue is deleted. IPCA is also a parameter-free approach. However, since each iteration involves SVD operation, the computational cost per iteration is $\mathcal{O}(k^2 d)$. In addition, IPCA is limited to processing one sample per update operation and extending the method to work in mini-batch mode is not possible. Despite the lack of convergence analysis of this method, it provides the best convergence results according to Cardot and Degras [CD18].

2.3.4 Similarity Matching using Hebbian and anti-Hebbian Learning

Recently, Pehlevan et al. proposed a method for finding the top k eigenvectors and their corresponding scores that works in both online and offline modes [PSC18]. Their technique is based on optimizing the similarity matching objective function as follows

$$\begin{aligned}\min_{Y\in\mathbb{R}^{k\times n}}\min_{W\in\mathbb{R}^{k\times d}}\max_{M\in\mathbb{R}^{k\times k}} L(M,Y,W) =& \mathrm{Tr}\left(-\frac{4}{n}X^TWY+\frac{2}{n}Y^TMY\right)\\ &+2\mathrm{Tr}\left(WW^T\right)-\mathrm{Tr}\left(M^TM\right).\end{aligned}$$

Here, $X\in\mathbb{R}^{d\times n}$ is the data matrix in column vector format, Y corresponds to scores (projection values) of samples onto the top k eigenvectors, $W=\frac{1}{n}XY^T$ contains the top k eigenvectors and $M=W^TW$. The optimization follows by applying gradient ascent-descent with respect to M and W respectively as follows

$$\begin{bmatrix}W & M\end{bmatrix}=\begin{bmatrix}W & M\end{bmatrix}+\begin{bmatrix}-\eta\frac{\partial}{\partial W}L & \frac{\eta}{\tau}\frac{\partial}{\partial M}L\end{bmatrix} \tag{2.22}$$

where η is the learning rate and $\tau>0$. Computing the optimal Y can be achieved as follows

$$Y=M^{-1}W^TX, \tag{2.23}$$

where the inversion of M exists since it is symmetric positive definite. The updates in the previous equations are achieved in an offline manner. In order to make the algorithm work in the online mode, the data matrix is replaced with the recent time step x_t and Y is replace by $y_t=M_t^{-1}W_t^Tx_t$ and

$$\begin{aligned}\begin{bmatrix}W_{t+1} & M_{t+1}\end{bmatrix}&=\begin{bmatrix}W_t & M_t\end{bmatrix}+\begin{bmatrix}-\eta_t\frac{\partial}{\partial W}L_t & \frac{\eta_t}{\tau}\frac{\partial}{\partial M}L_t\end{bmatrix}\\ &=\begin{bmatrix}W_t & M_t\end{bmatrix}\\ &\quad+\begin{bmatrix}2\eta_t\left(x_ty_t^T-W_t\right) & \frac{\eta_t}{\tau}\left(y_ty_t^T-M_t\right)\end{bmatrix},\end{aligned}$$

where $0<\eta_t<1$ is the learning rate and τ is set such that $0<\tau<1/2$ in order to converge surely to the optimal solution. The computational cost per update step is $\mathcal{O}(k^3+kd)$ due to the matrix inversion involved. In order to reduce computational cost of the matrix inversion

M_t^{-1}, y_t is computed iteratively using Hebbian and anti-Hebbian neurons according to

$$y_t^{i+1} = y_t^i + \gamma \left(W_t x_t - M_t y_t^i \right),$$

however, the number of updates required to arrive to optimal solution was not investigated by the authors. Nevertheless, the choice of appropriate γ values was not discussed.

2.3.5 Computing Multiple Eigenvectors

Many streaming PCA methods are tested for computing only one eigenvector (the most significant one). Computing multiple eigenvectors is also important, especially when most of the data variance is not covered by the first eigenvector. There are many possible ways to extend streaming PCA to deal with multiple eigenvectors. The Gram-Schmidt orthogonalization [Gra83; Sch07] is one common approach where after applying the update formula to k vectors, initialized differently and randomly, this scheme is performed. The main downside of this method is that its computational cost is $\mathcal{O}(k^2 d)$. This may significantly increase the runtime, especially when dealing with a large number of eigenvectors k. An alternative method is to apply the deflation process. In this case, the eigenvector W^j is updated based on a deflated sample which results by replacing the original input sample x by $x - \sum_{k=1}^{j-1} W^j (W^j)^T x$. Hence, eigenvectors are updated based on their descending order. While the cost is $\mathcal{O}(kd)$, one can note that this approach is sequential and may suffer round-off errors. One problem that may also occur when computing multiple eigenvectors is that the update rule might converge to a rotated version of the eigenspace. Recall that the philosophy of the holistic representation is to describe any sample as a combination of holistic features. The weightings of such features are what distinguishes between different categories. Since the eigenvectors derived using PCA are well-defined features, it is rather very important to ensure that the produced streaming solution does not correspond to a span of the eigenspace. However, verifying whether a solution does not suffer from such a problem is hard, specifically when computing a large number of eigenvectors.

2.3.6 Convergence Evaluation

In terms of convergence evaluation for streaming PCA methods, there are two strategies used in the literature. The first strategy is to analyze the convergence rates theoretically. The main focus for many of such theoretical papers is to derive a fast-converging streaming PCA method whose hyperparameters do not depend on the eigengap. Such methods are referred to as eigengap-free methods [AL17]. The second evaluation strategy is to study the convergence empirically using benchmark generative models and datasets. One commonly used benchmark is the spiked covariance model where samples are synthesized from unknown basis vectors that are generated randomly [MCJ13]. In terms of benchmark datasets, many papers study the convergence on benchmark machine learning datasets such as the MNIST dataset [Aro+12; ACS13; Sha15]. In addition, face datasets are also commonly used for evaluation [WZH03; CD18; BDF13]. For empirical evaluations, it is important to choose a suitable metric for measuring the convergence to the optimal eigenspace. Shamir proposed the following metric

$$\text{convergencece}\,(V^{*}, W_t) = \log_{10}\left(1 - \frac{\|X^T W_t\|_F^2}{\|X^T V^{*}\|_F^2}\right) \tag{2.24}$$

where $\|.\|_F^2$ is the squared Frobenius norm and $V^{*} \in \mathbb{R}^{d\times k}$ corresponds to the optimal solution and W_t is the eigenvectors estimation at time t [Sha15]. This function measures the percentage of variance maintained by the estimated eigenvectors to the variance maintained by the optimal eigenspace. Assume that the value returned by the function is -4, this means that the estimated eigenvectors maintained 0.9999 of the variance maintained by the optimal eigenspace. The main advantage of this metric is that it is very precise in comparing the convergence rates between two or more methods. The main limitation of this metric is that it requires the optimal solution which may not be possible to compute when dealing with very large-scale datasets as we will see in this thesis. For such scenarios, the explained variance may be used, which provides the percentage of total variance explained by each eigenvector. Another measure that can be used is the mean-squared-error (MSE) between reconstructed samples using the eigenvectors estimation and the original samples. For large-scale image datasets, the Structured Similarity (SSIM) can be used instead of MSE which is defined as follows

$$\text{SSIM}\,(x, y) = \frac{(2\mu_x\mu_y + c_1)(\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}, \tag{2.25}$$

where μ refers to the mean, σ^2 is the variance, σ_{xy} is the covariance between x and y and c_1 and c_2 are two parameters used for stabilizing the division. In this thesis, we will evaluate streaming PCA methods using the empirical evaluation strategy. We will use the spiked covariance model and other benchmark datasets, including MNIST, fashion-MNIST, CIFAR-10, Labeled-faces-in-the-wild (LFW) and Celebrity Attributes (CelebA) face datasets for evaluation. If the dataset is small enough, the metric defined in 2.24 will be used. For large scale datasets, we will use MSE and SSIM. It is also worth mentioning that, to the best of my knowledge, streaming PCA has been investigated only for datasets where samples are independent and identically distributed. In this thesis, we will also investigate time-varying systems which do not obey such an assumption. Hence, we find that the empirical evaluation strategy may be more appropriate for studying different types of datasets.

2.4 Generalizations

In this section, we will discuss some well-known generalizations to the standard PCA model. These models mainly extend the definition of the standard approach to deal with non-linear data. Such generalizations have inspired many of the techniques developed in this thesis. While such generalizations have many advantages in comparison to the standard approach, they suffer several limitations that are discussed in this section.

2.4.1 PCA and Neural Networks

The relation between PCA and neural networks was subject to a long-standing investigation. This link arose back in 1949 when Donald Hebb in his book *The Organization of Behaviour* [Heb49] had postulated that "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cats firing B, is increased". According to Lowell and Singer [LS92], this can be summarized as "neurons wire together if they fire together". In other words, this tells us that the synaptic weights in neurons are a measure of the correlations between their activations. Such synaptic weights correspond the connection strength between different neurons. If the correlation is positive, then the connection strength between the neurons increases and vice versa. Mathematically, this can be expressed ass follows

$$w_{t+1} = w_t + \eta_t x_t y_t \tag{2.26}$$

where η_t is the learning rate, $x_t \in \mathbb{R}^d$ describes the excitation on neuron at time t received from other neurons and $y_t \in R$ corresponds to the output (activation) of the neuron. In the linear sense, the activation function is written as $y_t = x_t^T w_t$ and the Hebbian rule becomes

$$w_{t+1} = w_t + \eta_t x_t x_t^T w_t. \tag{2.27}$$

It can be proven that the dynamics of this update rule moves towards the direction of the first significant eigenvector [DK96]. However, this rule is unstable as the magnitude of the vector increases to infinity with time. Hence, a simple remedy is to normalize the vector after each update which is known as the normalized Hebbian rule defined in Eq. 2.16. In the literature, this update rule is mistakenly referred to as Oja's rule. In fact, what Oja proposed in his paper [Oja82] is a linearized version of the normalized Hebbian rule described as follows

$$w_{t+1} = w_t + \eta_t \left(y_t x_t - y_t^2 w_t \right). \tag{2.28}$$

One can derive this formula by employing Taylor expansion to the normalized rule and neglecting higher-order terms. The main advantage of using this formula is the fact that it does not require computing the normalization step and hence is computationally cheaper. However, due to the fact that the normalized Hebbian rule is more common and easier to generalize for computing multiple eigenvectors, whenever referring to Oja's rule in this thesis, we consider the normalized Hebbian rule. Nevertheless, the convergence rate and dynamics of both schemes were found to be very similar.

As the field of neural networks evolved specifically after the proposition of the back-propagation learning algorithm, an interesting question raised by Bourlard and Kamp [BK88] concerned the relation between the solutions obtained using auto-association multi-layer perceptron and PCA and whether such a solution could lead to advantages over PCA. A two-layer feed-forward neural network with one bottleneck hidden layer was trained using back-propagation algorithm to generate the same input samples with a low number of hidden units (much lower that the sample dimensionality). It was found that even when using non-linear sinusoidal activations in the hidden layer, the weights of the network span the PCA eigenspace. Furthermore, the network was found to be subject to local minima problems leading to higher reconstruction errors in comparison to the deterministic PCA. Hinton and Salakhutdinov [HS06] studied the performance of multi-layer feed-forward neural network with multiple bottlenecks hidden layers of non-linear activations. Using only 30 neurons in the central hidden layer, the network was able to outperform the standard PCA when compared to the reconstruction error of the significant 30 eigenvectors. This model shed light on a new generation of multi-layer perceptron known as Autoencoders. Autoencoders are a special type of back-propagation Neural Networks that are trained to produce the same input samples in its output layer via a reduced representation of hidden units. The main characteristic of autoencoders is to find a reduced representation of the input dataset non-linearly.

An Introduction to the Back-propagation Learning Algorithm

In this section, we will describe the structure of multi-layer perceptron and how to apply the back-propagation learning algorithm which was proposed by Werbos in [Wer74]. The main reason for introducing the back-propagation algorithm is that it is the standard approach for training feed-forward neural networks and deep learning models. In this thesis, we will discuss deep learning in more detail in Chapter 5. An MLP model consists of one input layer, one or more hidden layers and one output layer. One can think of the output layer as a perception unit. Each unit in an MLP is called a *perceptron*. Each perceptron consists of (a) one or more weighted input units (synaptic weights), (b) an activation function f and (c) a single output unit. Figure 2.1 shows an example of multi-layer perceptrons consisting of n_0 input units in the input layer, n_1 perceptrons in the first hidden layer, n_2 perceptrons in the second hidden layer and n_3 output units in the output layer. Let us consider a set of m training samples $\{\Gamma_i \in \mathbb{R}^{n_0}\}_{i=1}^{m}$ and their corresponding targets $\{T_i \in \mathbb{R}^{n_3}\}_{i=1}^{m}$. We define the outputs of the first hidden layer as

$$X_{out}^{(1)} = \begin{bmatrix} x_{out,1}^{(1)} \\ \vdots \\ x_{out,n_1}^{(1)} \end{bmatrix} = \begin{bmatrix} f\left(v_1^{(1)}\right) \\ \vdots \\ f\left(v_{n_1}^{(1)}\right) \end{bmatrix}, \tag{2.29}$$

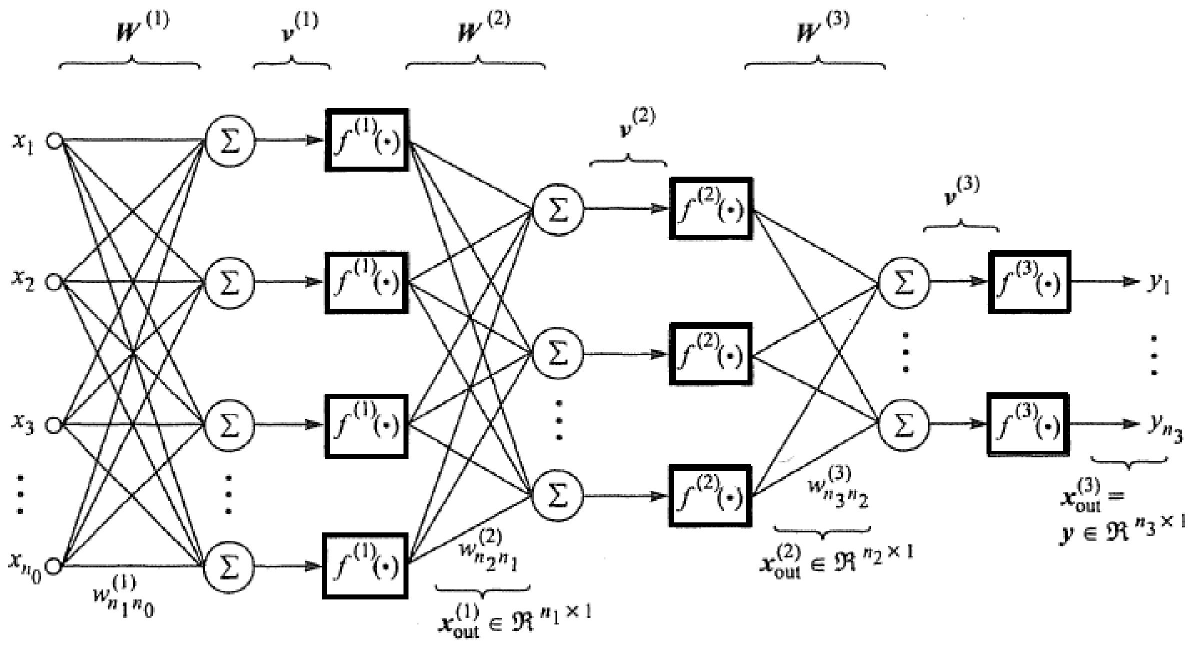


Figure 2.1: An example of a fully-connected MLP.

where $v_j^{(1)} = \sum_{i=1}^{n_0} x_i w_{i,j}^{(1)} + b^{(1)}$ and $b^{(1)}$ is a bias term. The outputs of the remaining layers are defined as

$$X_{out}^{(q)} = \begin{bmatrix} x_{out,1}^{(q)} \\ \vdots \\ x_{out,n_q}^{(q)} \end{bmatrix} = \begin{bmatrix} f\left(v_1^{(q)}\right) \\ \vdots \\ f\left(v_{n_q}^{(q)}\right) \end{bmatrix}, \; q = 2,\ 3, \tag{2.30}$$

where $v_j^{(q)} = \sum_{i=1}^{n_{q-1}} x_{out,i}^{(q-1)} w_{i,j}^{(q)} + b^{(q)}$. There are many options for defining the activation function f. One choice is to use the sigmoid function defined as follows

$$f(x) = \frac{1}{1 + \exp(-x)}.$$

Other options include the rectified linear unit (ReLU) defined as $f(x) = \max(0, x)$. Training such a network requires a number of training epochs where each epoch corresponds to a single data pass. The basic idea of the back-propagation algorithm is to adjust the weights of the network for each epoch until the error function converges to a local minima. The error function, also known as the loss function, measures how well the network learned the task on hand. One commonly used error function is the mean squared error (MSE) defined as

$$E = \frac{1}{m} \sum_{s=1}^{m} E_s, \tag{2.31}$$

where

$$E_s = \frac{1}{2}\left(T_s - X_{out}^{(3)}(\Gamma_s)\right)^T \left(T_s - X_{out}^{(3)}(\Gamma_s)\right) = \frac{1}{2}\sum_{h=1}^{n_3}\left(T_{s,h} - x_{out,h}^{(3)}(\Gamma_s)\right)^2$$

For classification tasks, the cross-entropy error function is more commonly used. In this example, we will use MSE as the error function. The goal of the BP algorithm is to adjust the network's weights such that the value of the error function is minimized. Before teaching the MLP, the initial weights are chosen randomly. Optimizing the error function can be achieved using the stochastic gradient descent or its variants. In this case, the gradient descent update rule is applied on a mini-batch of the training data that are visited only once during a single epoch. The choice of the mini-batch size depends on the training data. In this example, we will consider a mini-batch of size 1 for simplicity. Hence, applying the stochastic gradient descent will lead to

$$\triangle w_{i,j}^{(q)} = -\eta \frac{\partial E_s}{\partial w_{i,j}^{(q)}}, \tag{2.32}$$

where q denotes the layer number and $\eta > 0$ is the learning rate. We will begin by deriving $\triangle w_{i,j}^{(3)}$ for the output layer. Using the chain rule, the update rule can be rewritten as follows

$$\triangle w_{i,j}^{(3)} = -\eta \frac{\partial E_s}{\partial v_j^{(3)}} \frac{\partial v_j^{(3)}}{\partial w_{i,j}^{(3)}}. \tag{2.33}$$

The first factor can be written as

$$\frac{\partial E_s}{\partial v_j^{(3)}} = \frac{\partial}{\partial v_j^{(3)}}\left[\frac{1}{2}\sum_{h=1}^{n_3}\left(T_{s,h} - x_{out,h}^{(3)}\right)^2\right] = -\left[T_{s,j} - x_{out,j}^{(3)}\right] f'(v_j^{(3)}), \tag{2.34}$$

and for the second factor

$$\frac{\partial v_j^{(3)}}{\partial w_{i,j}^{(3)}} = \frac{\partial}{\partial w_{i,j}^{(3)}}\left[\sum_{i=1}^{n_2} x_{out,i}^{(2)} w_{i,j}^{(3)}\right] = x_{out,i}^{(2)}. \tag{2.35}$$

Combining the two derivatives and denoting $\delta_j^{(3)} = \left[T_{s,j} - x_{out,j}^{(3)}\right] f'(v_j^{(3)})$, we get

$$\triangle w_{i,j}^{(3)} = \eta x_{out,i}^{(2)} \delta_j^{(3)}, \tag{2.36}$$

or

$$w_{i,j}^{(3)}(k+1) = w_{i,j}^{(3)}(k) + \eta x_{out,i}^{(2)} \delta_j^{(3)}. \tag{2.37}$$

The update equation for the hidden layers of the network can be derived in the same manner. Applying the gradient descent approach to the neurons of the second hidden layer, we get

$$\triangle w_{i,j}^{(2)} = -\eta \frac{\partial E_s}{\partial w_{i,j}^{(2)}} = -\eta \frac{\partial E_s}{\partial v_j^{(2)}} \frac{\partial v_j^{(2)}}{\partial w_{i,j}^{(2)}}. \tag{2.38}$$

The second factor in the right hand side can be evaluated exactly the same as for the output layer. Hence

$$\frac{\partial v_j^{(2)}}{\partial w_{i,j}^{(2)}} = \frac{\partial}{\partial w_{i,j}^{(2)}} \left[\sum_{i=1}^{n_1} x_{out,i}^{(1)} w_{i,j}^{(2)} \right] = x_{out,i}^{(1)}. \tag{2.39}$$

However, evaluating the first factor is more complicated. To proceed, we apply the chain rule

$$\begin{aligned}
\frac{\partial E_s}{\partial v_j^{(2)}} &= \frac{\partial E_s}{\partial x_{out,j}^{(2)}} \frac{\partial x_{out,j}^{(2)}}{\partial v_j^{(2)}} \\
&= \frac{\partial}{\partial x_{out,j}^{(2)}} \left\{ \frac{1}{2} \sum_{h=1}^{n_3} \left[T_{s,h} - f\left(\sum_{p=1}^{n_2} w_{s,p}^{(3)} x_{out,p}^{(2)} \right) \right]^2 \right\} \frac{\partial x_{out,j}^{(2)}}{\partial v_j^{(2)}} \\
&= -\left\{ \sum_{h=1}^{n_3} \left[T_{s,h} - x_{out,h}^{(3)} \right] f'(v_h^{(3)}) w_{h,j}^{(3)} \right\} f'(v_j^{(2)}) \\
&= -\left(\sum_{h=1}^{n_3} \delta_h^{(3)} w_{h,j}^{(3)} \right) f'(v_j^{(2)}),
\end{aligned} \tag{2.40}$$

since $\delta_h^{(3)} = \left[T_{s,h} - x_{out,h}^{(3)} \right] f'(v_h^{(3)})$ are calculated already from the output layer and hence are fixed. Combining the two factors and denoting $\delta_j^{(2)} = \left(\sum_{h=1}^{n_3} \delta_h^{(3)} w_{h,j}^{(3)} \right) f'(v_j^{(2)})$, one gets the update equation

$$w_{i,j}^{(2)}(k+1) = w_{i,j}^{(2)}(k) + \eta x_{out,i}^{(1)} \delta_j^{(2)}. \tag{2.41}$$

From eq. (2.37) and (2.41), we can define the general update formula as follows:

$$w_{i,j}^{(q)}(k+1) = w_{i,j}^{(q)}(k) + \eta x_{out,i}^{(q-1)} \delta_j^{(q)},$$

where

$$\delta_j^{(q)} = \left[T_{s,j} - x_{out,j}^{(q)} \right] f'(v_j^{(q)})$$

for the output layer and

$$\delta_j^{(q)} = \left(\sum_{h=1}^{n_{q+1}} \delta_h^{(q+1)} w_{h,j}^{(q+1)} \right) f'(v_j^{(q)})$$

for the hidden layers. This process will be repeated until the error function reaches a predetermined threshold or when the maximum number of epochs is reached.

2.4.2 Subspace Clustering

In this subsection, we present one of the most recent generalizations to PCA, which is referred to as subspace clustering. Such a generalization has inspired many of the techniques developed in Chapter 4. The main idea of subspace clustering is to compute a set of subspaces where each subspace optimally fits a subset of samples in the dataset [Vid11]. A subspace is defined in terms of its mean point, basis vectors and projection values. Subspace clustering has recently become an active research area as an alternative to the traditional cluster analysis where samples are clustered based on their centroids. Such a methodology is particularly useful in case of extremely high dimensional data. More formally, consider a set of n data points that lie in d-dimensional space $\left\{x_j \in \mathbb{R}^d\right\}_{j=1}^{n}$, subspace clustering assumes that such samples are generated from a union of $n_s > 1$ subspaces. Each subspace is defined as follows

$$S_i = \left\{x \in \mathbb{R}^d \mid x = \bar{x}_i + U_i y\right\}, \quad i = 1, \dots, n_s, \tag{2.42}$$

where $\bar{x}_i$ is a point in subspace S_i, $U_i \in \mathbb{R}^{d\times k_i}$ corresponds to the basis vectors and $y = U_i^T x \in \mathbb{R}^{k_i}$ is a low dimensional representation of pint x. The subspace clustering problem can be formulated as follows: Given a dataset $X \in \mathbb{R}^{d\times n}$, we need to find the number of subspaces n_s, their dimensions $\{k_i\}_{i=1}^{n_s}$, their basis vectors $\left\{U_i \in \mathbb{R}^{d\times k_i}\right\}_{i=1}^{n_s}$, the points $\left\{\bar{x}_i \in \mathbb{R}^d\right\}_{i=1}^{n_s}$ and the segmentation of data samples according to the subspaces. Clearly, if the number of subspaces is equal to one, the problem is simplified to the standard PCA problem. The main challenge in this problem is that for n data points, there can be 2^n subgroups. Hence, optimally mapping the samples to different subspaces is rather challenging. Investigating different mapping strategies has been widely explored in the literature.

2.4.3 Kernel PCA

The standard PCA gives a linear description of the distribution of the input data using orthogonal eigenvectors (Principal Components). However, in many cases, the data may not be well-described using standard PCA due to its non-linear nature. The basic idea of kernel principal component analysis (kPCA), proposed in 1998 by Schölkopf et al. [SSM98], is to transform the input data into a higher-dimensional space, which is usually called the feature space, using a non-linear implicit transformation Ψ and then extract the kernel eigenvectors from the feature space using the kernel trick discussed below.

Suppose again we have n samples x_i, $i = 1, \dots n$, in $\mathbb{R}^d$. Using kPCA, these samples are projected onto a higher dimensional space (feature space) using non-linear transformation: $\Psi : \mathbb{R}^d \to \mathbb{R}^f$, $f \gg d$. Assuming that the samples are centered in the feature space $\sum_{i=1}^{n} \Psi(x_i) = 0$, the $f \times f$ covariance matrix of these samples in the feature space can be defined as

$$C^{\Psi} = \frac{1}{n-1}\sum_{i=1}^{n}\Psi(x_i)\Psi(x_i)^T. \tag{2.43}$$

The corresponding eigenvalue problem becomes

$$(C^{\Psi} - \lambda^{\Psi} I)v^{\Psi} = 0, \tag{2.44}$$

where $\lambda^{\Psi} > 0$ and $v^{\Psi} \in \mathbb{R}^f$ are a non-negative eigenvalue and its corresponding eigenvector in the feature space respectively. Eq. (2.44) can be rewritten as

$$\left\{\frac{1}{n-1}\sum_{i=1}^{n}\Psi(x_i)\Psi(x_i)^T\right\}v^{\Psi} = \frac{1}{n-1}\sum_{i=1}^{n}\Psi(x_i)\left\{\Psi(x_i)^T v^{\Psi}\right\} = \lambda^{\Psi}v^{\Psi}. \tag{2.45}$$

By denoting $\alpha_i = \left\{\frac{1}{\lambda^{\Psi}(n-1)}\Psi(x_i)^T v^{\Psi}\right\} \in \mathbb{R}$, from eq. (2.45), we have

$$\sum_{i=1}^{n}\Psi(x_i)\alpha_i = v^{\Psi}. \tag{2.46}$$

This means that all eigenvectors corresponding to the non-zero eigenvalues lie in the span of $\Psi(x_1), \cdots, \Psi(x_n)$. By substituting eq. (2.46) in eq. (2.45), we get

$$\left\{\frac{1}{n-1}\sum_{i=1}^{n}\Psi(x_i)\Psi(x_i)^T\left\{\sum_{j=1}^{n}\Psi(x_j)\alpha_j\right\}\right\} = \lambda^{\Psi}\left\{\sum_{i=1}^{n}\Psi(x_i)\alpha_i\right\}. \tag{2.47}$$

The kernel trick

Since one doesn't need to know Ψ, the kernel trick uses the duality property to solve the eigenvalue problem in eq. (2.47) without defining Ψ explicitly. This is done by defining a kernel function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ that, under some conditions (Mercer's conditions), satisfies

$$k(x, y) = \Psi(x)^T\Psi(y)$$

for any x, $y \in \mathbb{R}^d$. The choice of kernel function depends on the distribution of input samples. Commonly used kernel functions include the polynomial kernel function of order d defined as $k(x, y) = \left(x^T y + c\right)^d$ with c being a constant and the Gaussian kernel function $k(x, y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$.

By multiplying each side of eq. (2.47) by $\Psi(x_q)^T$, $q = 1\cdots n$, this becomes

$$\left\{\frac{1}{n-1}\sum_{i=1}^{n}k(x_q, x_i)\left\{\sum_{j=1}^{n}k(x_i, x_j)\alpha_j\right\}\right\} = \lambda^{\Psi}\left\{\sum_{i=1}^{n}k(x_q, x_i)\alpha_i\right\}. \tag{2.48}$$

This can be rewritten in the matrix form as follows:

$$K^2 a = (n-1)\lambda^{\Psi} K a, \tag{2.49}$$

where $a = [\alpha_1\ \alpha_2\ \dots\ \alpha_n]^T$ is the dual eigenvactor and

$$K = \begin{bmatrix} k(x_1,x_1) & \cdots & k(x_1,x_n) \\ \vdots & \ddots & \vdots \\ k(x_n,x_1) & \cdots & k(x_n,x_n) \end{bmatrix} = \begin{bmatrix} \Psi(x_1)^T\Psi(x_1) & \cdots & \Psi(x_1)^T\Psi(x_n) \\ \vdots & \ddots & \vdots \\ \Psi(x_n)^T\Psi(x_1) & \cdots & \Psi(x_n)^T\Psi(x_n) \end{bmatrix} \tag{2.50}$$

is the kernel matrix, where each (i,j)th element corresponds to the kernel function of x_i and x_j. Assuming that K is invertible, eq. (2.49) can be simplified to

$$\left(\frac{1}{n-1}K - \lambda^{\Psi} I\right) a = 0. \tag{2.51}$$

This trick eliminates the direct computation of the covariance matrix C^{Ψ} and hence the explicit calculation of the non-linear transformation Ψ. Instead, one only needs to compute the kernel matrix K and then solve the eigenvalue problem in eq. (2.51). The projection value of an input sample s onto the kernel eigenvector $v_j^{\Psi} j = 1 \dots p$ in the feature space can be calculated as follows

$$\rho^{\Psi} = \left(v_j^{\Psi}\right)^T \Psi(s) = \sum_{i=1}^{n} \alpha_i^j k(x_i,\ s). \tag{2.52}$$

The steps of the KPCA can be summarized as follows:

1. Calculate the kernel matrix K using eq. (2.50),
2. Solve the eigenvalue problem in eq. (2.51),
3. Project the samples onto the kernel eigenspace using eq. (2.52).

One can note how the kernel trick simplifies the computation of the kernel eigenspace using a few simple steps. One point to consider when applying kPCA is that the kernel function and its hyperparameters need to be chosen carefully. In addition, reconstructing samples from the high dimensional Hilbert space is computationally expensive. Studying kernel PCA in the streaming mode has been investigated in [Hon11] where Oja's rule was adapted to work with the kernel trick.

2.5 Analogy with Compression Models

In this section, we discuss the discrete cosine transform which is widely employed in image and video compression standards such as JPEG and MPEG. One interesting point in these compression standards is the fact the DCT is performed in a block-wise manner where the image is subdivided into blocks and the DCT is applied to each block separately. This discussion will raise the following questions: What are the advantages of applying basis function representations in the partitioning manner instead of the holistic way? Will using PCA in the partitioning manner be advantageous compared to the standard approach and other basis function representations? These questions will be investigated further in this thesis in Chapter 4.

2.5.1 The Discrete Cosine Transform

The Discrete Cosine Transform is a Fourier-related basis function representation that is widely used for image compression. It was first introduced by Nasir Ahmed in 1974 [ANR74]. While there are several types of discrete cosine transform, the best known type is the DCT-II which is the one that by default the term DCT refers to. The main idea behind this technique is to express any input signal using a weighted sum of a few low-frequency cosine basis functions. Cosine basis functions are used instead of sine functions because they work better for different boundary conditions. The sine transform is only suitable when dealing with zero boundary conditions for which the cosine basis functions can also provide a good approximation. These cosine basis functions can also be defined in 2D space, making them capable of representing 2D images. The 2D DCT transforms an image A_{mn} of size $M \times N$ to the DCT space B_{pq} using the following formula

$$B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} \cos \frac{\pi(2m+1)p}{2M} \cos \frac{\pi(2n+1)q}{2N}$$

where

$$\alpha_p = \begin{cases} \frac{1}{\sqrt{M}}, & p = 0 \\ \sqrt{\frac{2}{M}}, & p > 0 \end{cases}$$

and

$$\alpha_q = \begin{cases} \frac{1}{\sqrt{N}}, & q = 0 \\ \sqrt{\frac{2}{N}}, & q > 0 \end{cases}$$

Figure 2.2 shows an example of a gray-scale image reconstructed using a few low-frequency 2D-DCT coefficients. It can be noted that coefficients corresponding to low frequencies have higher amplitude and vice versa. One main problem when using 2D-DCT for compression is the difficulty in prioritizing the basis functions in the 2D frequency space. In particular, what is the best order of basis functions for representing a set of samples? This becomes even more problematic when considering the representation of 3D volumetric data using 3D cosine basis functions. In addition, not all types of images are well-represented using the 2D-DCT basis functions. For instance, it has been reported that line drawing images give aliased results

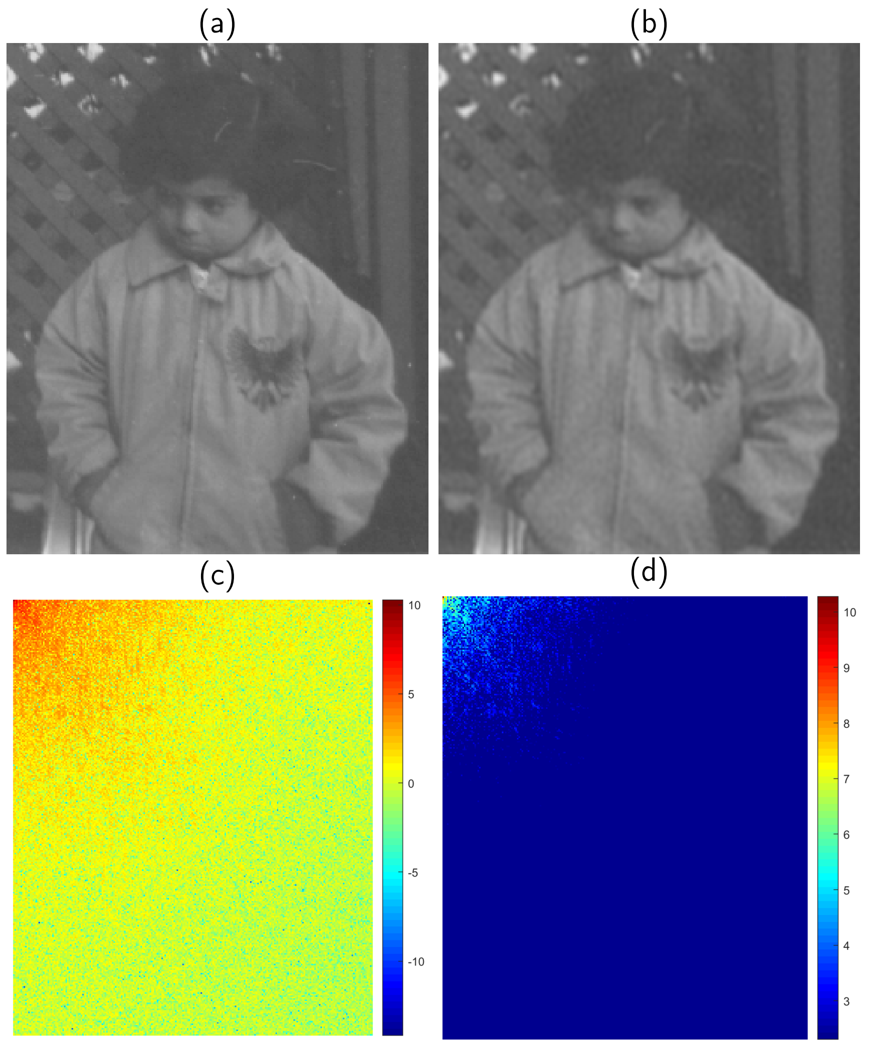


Figure 2.2: Reconstructing a gray-scale image using low-frequency DCT coefficients. (a) Original image, (b) Reconstructed image, (c) DCT coefficients of the original image represented using $\log_{10} \text{abs}(B_{pq})$ and (d) The coefficients used for reconstruction.

when reconstructed using DCT.

2.5.2 JPEG Compression

JPEG is an image compression method that is based on the 2D-DCT [Wal92]. Instead of using DCT as a holistic representation for the input image, the image is subdivided into smaller subblocks of size 8×8. Each block is then represented using its DCT coefficients. In order to achieve compression, only a few DCT coefficients are chosen. While there is no standard way of choosing the appropriate coefficients for each block, JPEG uses quantization-matrices (Q-matrices) for choosing the appropriate basis functions for each block [Kor08]. The Q-matrix is an 8×8 matrix consisting of divisors for the corresponding projection values. Projection values are divided by their divisors and then rounded in order to achieve quantization and compression by neglecting zero rounded resultants. There is no trivial way of defining the

Q-matrix. However, the Independent JPEG Group (IJG) tables are the most commonly used Q-matrices.

2.5.3 Analogy to PCA

Both PCA and DCT are basis function representations. However, the DCT provides a global representation that is independent of the input data distribution. On the other hand, PCA is a data-driven representation in which the basis functions are optimally computed to minimize the MSE when using a fixed number of the most significant eigenfunctions. Having that said, the computation of the eigenfunctions may be infeasible to find when dealing with high dimensional data. This problem will be further discussed in Chapter 4.

2.6 Eigenvalue Problems in Physical Models and Analogy to PCA

In this section, we will review some of the most well-known mathematical models that are applied for modelling time-varying physical phenomena. We will theoretically study the analogy between the PCA eigenvalue problem and some of these models in Chapter 6. Based on our theoretical analysis, we show that, for a wide range of physical phenomena, the eigenvectors derived using PCA are analogous to the physical analytic model.

2.6.1 The Helmholtz Equation

The Helmholtz equation is one of the most commonly used partial differential equations (PDEs) in Physics. It is basically an eigenvalue problem which has the following form

$$\nabla^2 A + k^2 A = 0, \tag{2.53}$$

where ∇^2 is the second order spatial derivative (Laplacian operator), k^2 is a positive constant and A is the amplitude. In the simplest case of 1D space, the solution to this equation is $A(x) = \alpha_0 e^{-ikx}$ where $i = \sqrt{-1}$ is the imaginary unit and α_0 is a constant whose value is determined by the initial conditions of the problem. Many physical problems involving PDE in space and time are simplified to this model by applying the technique of separation of variables as we will see later in this section.

2.6.2 Heat (Diffusion) Equation

The heat equation is a PDE that describes the distribution of heat in material over time. The heat equation is of historical importance since it is particularly the model that Fourier used to develop his well known Fourier Transform. It has the following general form

$$\nabla^2 A - \frac{1}{\alpha}\frac{\partial}{\partial t}A = 0, \tag{2.54}$$

where α is the thermal diffusivity of the studied material and A is the temperature function. The solution to this model is acquired by applying a technique called separation of variables. Using this technique, it is assumed that the solution is a multiplication of functions each involving only one variable. For instance, in 1D case we write

$$A(x,t) = X(x)\,T(t)\,.$$

Plugging this into the general equation, we get

$$\frac{T'}{\alpha T} = \frac{X''}{X}.$$

Since each side of the equation is independent of the other, both sides equal a constant $-\lambda$. Hence, we arrive to the following two ordinary differential equations (ODEs) which are of the form of eigenvalue problems

$$\frac{d}{dt}T + \alpha\lambda T = 0 \tag{2.55}$$

and

$$\frac{d^2}{dx^2}X + \lambda X = 0. \tag{2.56}$$

The general solutions to these ODEs are

$$T(t) = \beta_0 e^{-\lambda\alpha t} \tag{2.57}$$

and

$$X(x) = \beta_1 \sin\left(\sqrt{\lambda}x\right) + \beta_2 \cos\left(\sqrt{\lambda}x\right). \tag{2.58}$$

Assuming that we have the following initial and boundary conditions

$$A(0,t) = A(L,t) = 0, \qquad A(x,0) = f(x)\,, \tag{2.59}$$

where $f(x)$ is a given function describing the initial heat distribution. One can find that $\beta_2 = 0$ and that the eigenvalue satisfying these conditions will have the following possible solutions

$$\sqrt{\lambda} = \frac{n\pi}{L}; \quad n = 1,\ 2, \ldots, \infty.$$

Hence, the general solution becomes the sum of all possible eigenfunctions

$$A(x,t) = X(x)\cdot T(t) = \sum_{n=1}^{\infty} X_n(x)\,T_n(t) = \sum_{n=1}^{\infty} \varphi_n \sin\left(\frac{n\pi}{L}x\right) e^{-\left(\frac{n\pi}{L}\right)^2 \alpha t}. \tag{2.60}$$

It is worth mentioning that these eigenfunctions form an orthogonal bases where φ_n defines the score of $f(x)$ in that space which is basically given by the inner product between $f(x)$ and each eigenfunction as follows

$$\varphi_n = \frac{2}{L}\int_0^L f(x)\sin\left(\frac{n\pi}{L}x\right)dx.$$

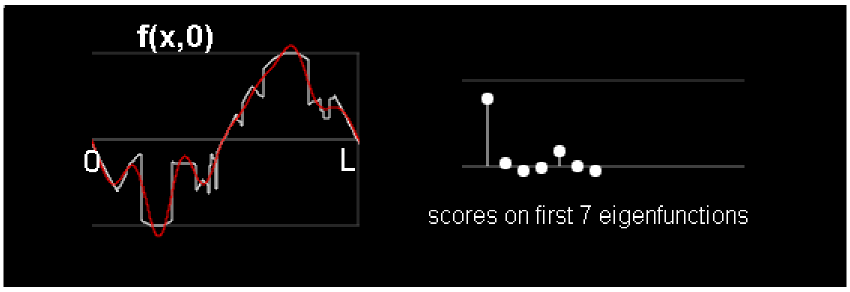


Figure 2.3: Approximating $f(x,0)$ using few terms of Fourier series.

The interesting part of this model is to note how this series approximates the initial distribution $f(x,0)$ using only few terms of the eigenfunctions. For instance, Fig. 2.3 shows how this function can be approximated using only first seven terms of the series. This is somehow similar to the way PCA can approximate the input samples using a combination of few eigenvectors. However, in the Fourier technique, this eigenspace is constituted of orthogonal bases of sinusoidal eigenfunctions.

2.6.3 The Wave Equation

The wave equation is one of the most important PDEs in Physics. It is widely used in physical applications including acoustics, electromagnetism and fluid dynamics. The general form of this model is given below

$$\nabla^2 A - \frac{1}{c^2}\frac{\partial^2}{\partial t^2}A = 0. \tag{2.61}$$

The general solution can be acquired by applying separation of variables to the above equation by setting $A(x,t) = X(x)\,T(t)$. This will simplify the model to a Helmholtz eigenvalue problem. It is also worth mentioning that the solution to this equation is defined by the term eigenmode or normal mode which is a solution that oscillates in time by letting the temporal part of the wave function $T(t) = e^{-iwt}$ where w describes the frequency of such oscillation.

Now, we will derive the general equation using Hooke's law for spring motion. This is an important way of understanding the wave model, which will be used later in this study. According to Hooke's law, a helical spring attached to the wall will suffer a restoring force when displaced from its equilibrium state (either when compressed or stretched). The restoring force acting on the free end of the spring is given by the following formula

$$F = kx, \tag{2.62}$$

where k is a constant describing the stiffness of the spring and x is the distance of the free end from its equilibrium position. Such a restoring force will cause the free end to oscillate around its equilibrium position. In other words, the force will change direction once the spring change from a stretched state to a compressed state and vice versa. Now, we will generalize this case to an array of objects of equal masses n connected using massless springs each of equilibrium length h and stiffness of k as shown in Fig. 2.4. $A(x,h)$ is a time dependent function describing the displacement of an object from its original position. According to Hooke's law, a force exerted on the object located on $x+h$ will be

$$\begin{aligned} F &= k\left[A(x+2h,t) - A(x+h,t)\right] - k\left[A(x+h,t) - A(x,t)\right] \\ &= k\left[A(x+2h,t) - 2A(x+h,t) + A(x,t)\right]. \end{aligned} \tag{2.63}$$

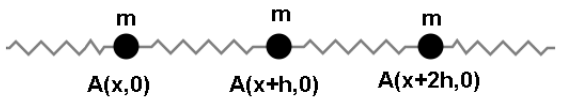


Figure 2.4: Array of 3 weights connected with spring in 1D settings.

But according to Newton's law, the force is given by the mass of the object times its acceleration $F = ma$ and since the acceleration is the second time derivative of the position function, one can write the above equation as follows

$$\begin{aligned} ma &= k\left[A(x+2h,t) - 2A(x+h,t) + A(x,t)\right] \\ \Rightarrow m\frac{\partial^2}{\partial t^2}A(x+h,t) &= k\left[A(x+2h,t) - 2A(x+h,t) + A(x,t)\right] \\ \Rightarrow \frac{\partial^2}{\partial t^2}A(x+h,t) &= \frac{k}{m}\left[A(x+2h,t) - 2A(x+h,t) + A(x,t)\right]. \end{aligned} \tag{2.64}$$

Now assuming an N body array , the total mass becomes $M = Nm$, the spring constant $K = \frac{k}{N}$ and the total length $L = Nh$. By noting that $\frac{k}{m} = \frac{L^2K}{Mh^2}$ we can rewrite the above equation as follows

$$\frac{\partial^2}{\partial t^2}A(x+h,t) = \frac{L^2K}{M}\frac{\left[A(x+2h,t) - 2A(x+h,t) + A(x,t)\right]}{h^2}. \tag{2.65}$$

By making $N \to \infty$ and $h \to 0$ we arrive to the following PDE

$$\frac{\partial^2}{\partial t^2} A(x,t) = \frac{L^2 K}{M} \frac{\partial^2}{\partial x^2} A(x,t) \tag{2.66}$$

which is of the form of the wave equation. The analogy between PCA and the wave equation will be discussed in more detail in Chapter 6.

3 One-pass Data Modelling Using Mini-batch Streaming PCA

In this chapter, we address the streaming PCA problem, which concerns the computation of the eigenspace from a single data pass. We propose an acceleration scheme that significantly improves the convergence rates of a wide range of streaming PCA methods that are based on the stochastic gradient approximation (SGA). Such methods are considered the cheapest in computational cost compared to other streaming approaches. In order to further increase the speed-up, we study the convergence of such techniques under the mini-batch mode which processes a block of samples at each update step instead of updating the eigenspace based only on a single sample.

3.1 Introduction

Streaming PCA schemes are commonly used to address the high computational cost of the standard approach. The main aim of such methods is to arrive at an acceptable solution after only a single pass over the entire set of data samples. Since each sample is visited only once, such techniques are also appropriate for streaming scenarios. Most well-known streaming PCA schemes are based on the stochastic gradient approximation. Such paradigms involve a learning rate parameter for which a choice plays a critical impact on the convergence rate [BDF13]. Choosing inappropriate learning rates may result in a very slow convergence. In fact, it was found that the optimal learning rate for such approaches requires knowing the eigengap $\Delta = \lambda_1 - \lambda_2$ which is impractical for streaming cases as it requires a preprocessing data pass. Another general problem that applies to all streaming algorithms concerns their sequential nature. When computing multiple eigenvectors of very high dimensional data, this leads to overhead computations due to the expensive orthogonalization processes.

Considering a dataset of stationary distribution $X = [x_1, x_2, \dots, x_n] \in \mathbb{R}^{d \times n}$ where d is the total number of dimensions (attributes), our goal in this chapter is to investigate effective ways for finding the top k eigenvectors from a single data pass without any prior knowledge about the distribution of the input samples. We propose an acceleration scheme for streaming PCA algorithms that are based on the stochastic gradient approximation. Particularly, our approach aims to accelerate such algorithms in the mini-batch mode in which each update step processes multiple samples of the dataset that are visited only once. Such a mode of computation has many advantages over the conventional fully-online streaming approach to PCA which processes only one sample per update. Mainly, processing samples in a mini-batch manner increases the speed because of the reduction in the number of expensive orthogonalization steps in addition to increasing parallelism. This chapter also provides a thorough empirical evaluation of state-of-the-art streaming PCA methods.

Main Contributions

The main contributions of this chapter are listed below:

- We propose a convergence acceleration scheme for SGA-based methods that work in the mini-batch setting.
- Our scheme works with smaller mini-batch sizes compared to the literature and hence is more suitable for large-scale datasets.
- We provide empirical evaluation strategies using the spiked-covariance model and other benchmark datasets and show that applying our acceleration enhanced convergence rates significantly outperforming state-of-the-art techniques.
- We show that when using our scheme for initializing the offline power iteration, the convergence rates were significantly enhanced.

3.2 Related Work

Amongst all streaming PCA approaches, the Stochastic Gradient Approximation (SGA) is one of the oldest attempts to address the online learning behaviour of PCA. Such approaches are based on the assumption that the covariance matrix can be approximated using an arbitrary sample x from the input dataset $\mathbb{E}\left(xx^T\right) = C$. The most well-known SGA techniques are the ones proposed by Krasulina [Kra69] and Oja [Oja82; OJE83]. One main limitation of these techniques is their performance sensitivity to different choices of learning rates, which may significantly affect the convergence rate. Considering that the learning rate has the form $\eta_t = c/t$, according to Balsubramani et al. [BDF13] the optimal performance of SGA methods is achieved when the initial learning rate constant $c = \frac{a}{\Delta}$ where $a > 1$ and $\Delta = \lambda_1 - \lambda_2$ is the eigengap. Hence, finding the optimal learning rate becomes infeasible in case of streaming scenarios as computing the eigengap requires a pre-processing data pass. Furthermore, this suggests that each individual eigenvector (when computing multiple $k > 1$ eigenvectors) should have its own learning rate. The computational cost of these algorithms is $\mathcal{O}(knd)$ FLOPs where k is the number of eigenvectors to be computed, n is the number of samples and d is the dimensionality. These techniques are considered the cheapest in terms of computational cost and space complexity. Other approaches that are based on SGA include the mini-batch (or block) power method proposed by Mitliagkas et al. [MCJ13] which applies the power iteration on mini-batches of the data that are visited only once. The main drawback of this method is that it may not converge in case of a very small mini-batch size B. It was proven that the convergence of this method is achieved when $B \geq \frac{n}{log(d)}$. Li et al. [LLL16] showed that the convergence of the mini-batch power method may be significantly improved when dynamically increasing the mini-batch sizes. Hardt and Price studied the convergence of the mini-batch power method in a more general context when data samples are not normally distributed [HP14]. De Sa et al. studied the convergence of state-of-the-art SGA methods after applying the momentum term for accelerating the convergence rate [Xu+18]. They showed that applying stochastic gradient descent with momentum to the mini-batch power method and variance-reduced power method (based on VR-PCA [Sha15]) gives better convergence rates compared to the original approaches. Candid Covariance-free Incremental PCA (CCIPCA) [WZH03] is another streaming PCA algorithm that has a significant advantage over Oja's method in the fact that it is parameter-free. Other parameter-free streaming PCA methods include Incremental PCA (IPCA), which is based on diagonalizing a reduced-order matrix of size $k \times k$ instead of dealing with the massive $d \times d$ covariance matrix [Aro+12]. The main downside of IPCA is that its complexity is $\mathcal{O}(k^2nd)$ which is more expensive compared to SGA-based methods. According to Cardot et al. [CD18] when comparing between, SGA, IPCA and CCIPCA, it was found that IPCA and CCIPA produced the best convergence results. Recently, Pehlevan et al. proposed a method for finding the top eigenvectors based on optimizing the similarity matching objective function. Their method works in both online and offline modes [PSC18]. We will refer to their method in this thesis as SM. One important point to highlight is that the performance of Oja, CCIPCA and SM in the mini-batch mode is a matter for further research.

3.3 An Acceleration Scheme for Mini-batch Streaming PCA

3.3.1 Problem Formulation and Intuition

Recall that the main goal of SGA-based streaming PCA methods is to optimize the PCA objective function after a single data pass. Recall also that the PCA objective is

$$\begin{aligned} \text{maximize} \quad & H(w) = w^T C w \\ \text{subject to} \quad & w^T w = 1, \end{aligned}$$

where C is the covariance matrix and w is the eigenvector. The main mechanism of such methods is to iteratively apply an update rule of the form $w_{t+1} = f(w_t, \tilde{X}_t)$ where $\tilde{X}_t = \{x_i\}_{i=Bt+1}^{B(t+1)}$ is a subset of the input dataset revealed at time t with cardinality B and $w_t \in \mathbb{R}^d$ is the eigenvector estimation at time t. We call this subset a block or mini-batch of samples of size B. For instance, in case of Oja's rule $f(w_t, \tilde{X}_t) = \frac{w_t + \eta_t \tilde{X}_t \tilde{X}_t^T w_t / B}{\left\| w_t + \eta_t \tilde{X}_t \tilde{X}_t^T w_t / B \right\|}$ and in case of the mini-batch (block) power method $f(w_t, \tilde{X}_t) = \frac{\tilde{X}_t \tilde{X}_t^T w_t}{\left\| \tilde{X}_t \tilde{X}_t^T w_t \right\|}$. Note that here we consider a mini-batch variant of Oja's method, where at each iteration we update the eigenvector based on a mini-batch of recent time-steps instead of using only the most recent one.

Since in the streaming PCA setting the covariance matrix C is unknown, one problem in such a setting is that it is challenging to know how well the streaming PCA method is converging. In other words, such methods do not provide any information about the convergence rate. The main idea of our acceleration is to define a measure that reflects how well the method is converging. The measure provides a bounded score between 0 and 1, where one corresponds to the state of convergence. Based on this measure, we accelerate the streaming PCA method such that the measure scores are optimized. In other words, we use such a measure to catalyze the streaming PCA in reaching the convergence state. Before defining our measure, let us discuss the primary state at which the streaming PCA converges. The convergence condition corresponds to a stable steady state where the difference between two successive updates becomes negligible. Assume that w_t and w_{t+1} are the two normalized eigenvector estimations at time t and $t+1$, the measure should return one if the two vectors are pointing either to the same or opposite directions and zero if the two vectors are orthogonal. This can be expressed as follows

$$G\left(w_{t+1}, w_t\right) = \left(w_{t+1}^T w_t\right)^2 = w_{t+1}^T w_t w_t^T w_{t+1}, \tag{3.1}$$

where w_t and w_{t+1} are normalized and lie in $\mathbb{R}^d$ space. In order to accelerate the convergence, we update the estimation w_{t+1} so that it will better satisfy the convergence condition. This can be achieved by directly applying the gradient ascent to G as follows

$$\begin{aligned}
\tilde{w} &= w_{t+1} + \frac{\alpha_t}{2} \frac{\partial}{\partial w_{t+1}} G\left(w_{t+1}, w_t\right) \\
&= w_{t+1} + \frac{\alpha_t}{2} \frac{\partial}{\partial w_{t+1}} \left(w_{t+1}^T w_t w_t^T w_{t+1}\right) \\
&= w_{t+1} + \alpha_t w_t w_t^T w_{t+1} \\
&= \left(I + \alpha_t w_t w_t^T\right) w_{t+1}
\end{aligned} \tag{3.2}$$

where α_t is the learning rate, the values for which will be discussed in the following subsection. Note that we derive the measure G with respect to w_{t+1} not w_t. This is to ensure that after each update, the eigenvector estimation is achieving better convergence. Hence one can arrive at the following update rule

$$\tilde{w}_{t+1} = \frac{w_{t+1} + \alpha_t w_t w_t^T w_{t+1}}{\left\| w_{t+1} + \alpha_t w_t w_t^T w_{t+1} \right\|}. \tag{3.3}$$

Note that here we consider computing only a single eigenvector, however, we can easily generalizing this technique to compute multiple eignvectors as will be discussed in the next subsection.

3.3.2 The Algorithm

We can summarize the main steps of our acceleration for the $k = 1$ case as follows

1. initialize w_1 at random from the unit sphere in $\mathbb{R}^d$.
2. for $t = 1, 2, \dots, n/B$
   (a) Receive the next mini-batch $\tilde{X}_t$.
   (b) Update $w_{t+1} = f(w_t, \tilde{X}_t)$ where f is the update rule of the SGA method.
   (c) Apply the acceleration scheme $\tilde{w}_{t+1} = \frac{w_{t+1} + \alpha_t w_t w_t^T w_{t+1}}{\left\| w_{t+1} + \alpha_t w_t w_t^T w_{t+1} \right\|}$.
   (d) Set $w_{t+1} = \tilde{w}_{t+1}$.

Our scheme can be easily generalized to compute multiple eigenvectors ($k > 1$ case) by defining $W_t \in \mathbb{R}^{d \times k}$ and replacing the normalization process by the Gram-Schmidt orthonormalization or a deflation process. Algorithm 3.1 shows the pseudo code of our acceleration when applied to the mini-batch power method for computing k eigenvectors. The normalization function in line 3 normalizes each column vector in the input matrix. Algorithm 3.2 shows the pseudo code of our acceleration when applied to Oja's method for computing k eigenvectors. It is also worth mentioning that there are two main situations in which the objective function can be maximized. The first situation is when subsequent updates are converging towards the optimal solution which is the desired behaviour that we wish to get. The second case is when subsequent eigenvectors are not significantly changing due to the very small step-size when accelerating Oja's method. Therefore when accelerating Oja's rule, it would be better to set the initial learning rate η_1 with a large value.

3.3.3 Strategies for Scheduling the Learning Rate α_t

We now investigate the appropriate scheduling strategies for the learning rate α_t. We find that using increasing scheduling strategies of the form $\alpha_t = \mathcal{O}(t)$ surprisingly gives the best convergence results. This might be due to the fact that as t increases W_t should provide a better estimation of the optimal solution. In other words, one can think of the learning rate in this case as a certainty factor of the convergence of W_t estimation. Figure 3.1

Algorithm 3.1: Accelerated Mini-batch Power Method (Accelerated Block Power)

```
Input: Mini-batch size $B$, dataset mini-batches $X = [\tilde{X}_1 \, \tilde{X}_2 \cdots \tilde{X}_{n/B}]$, Learning rate
       scheduler $\alpha_t$ and the number of eigenvectors to compute $k$
Output: The $k$ most significant eigenvectors of $X$
1 Initialize each column of $W_1$ randomly from the unit sphere in $\mathbb{R}^d$;
2 for $t = 1, 2, \ldots n/B$ do
3     $W_{t+1} = \text{Normalize}\left(\tilde{X}_t \tilde{X}_t^T W_t\right)$;
4     $\tilde{W} = W_{t+1} + \alpha_t W_t W_t^T W_{t+1}$;
5     $W_{t+1} = \text{Orthonormalize}\left(\tilde{W}\right)$;
6 end
```

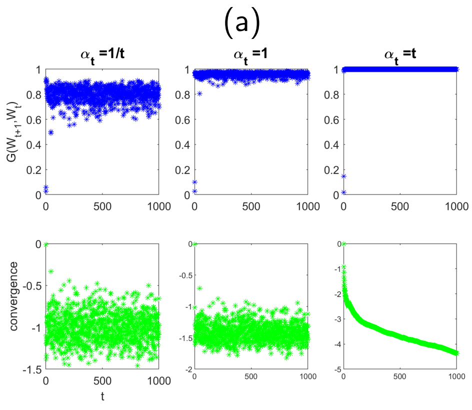
(a) shows the performance when accelerating the block power using decreasing $\alpha_t = 1/t$, constant $\alpha_t = 1$, and increasing $\alpha_t = t$ learning rates where it is obvious that increasing the learning rate provides the best performance. The use of increasing learning rates in the gradient-based optimization, while not usual, has been found to enhance the convergence rates in deep neural networks especially in cases where saddle points are present in the error surface [Smi17]. Comparisons of convergence rates for the block power and Oja's methods before and after applying our acceleration scheme are shown in Figure 3.1 (b). It is evident that using our scheme significantly enhances the convergence rates. In addition, one can note that the best convergence rates are achieved when the objective function reaches its optimal value of one. We will consider two stochastic strategies for scheduling the learning rate α_t. The first strategy that we propose is setting $\alpha_t = \frac{t}{1+cz_t}$ where c is some small constant (say 1) and $z_t \in [0, 1]$ is random variable of uniform distribution. The second strategy is based on setting $\alpha_t = \frac{t}{1+cz_t/t}$ where c in this case takes larger values depending on the sample size (say 1,000 in case of the number of samples is in the order of thousands). The main motivation for employing such stochastic schedulers is to allow for some adaptive behaviour for finding the optimal learning path. In the next section, we will test the technique using both strategies. In all the experiments in this chapter, we set $c = 1$ when using the first strategy and $c = 1,000$ when using the second scheduling strategy.

3.4 Performance Evaluation

In this section, we provide an empirical evaluation of streaming PCA algorithms based on the spiked covariance model and benchmark datasets. In general, evaluating streaming PCA approaches is challenging, especially when computing $k > 1$ eigenvectors. On the one hand, we need to measure the convergence to a global solution which can only be found by computing the traditional SVD or EVD in the batch mode. Hence, such measurement would not be possible if the dataset is too large to perform SVD. Also in many cases, the top eigenvectors may be very similar in terms of their eigenvalues leading to a phenomenon where the streaming

Algorithm 3.2: Accelerated Oja's Mathod (Accelerated Oja)

```
Input: Mini-batch size $B$, dataset mini-batches $X = [\tilde{X}_1\,\tilde{X}_2 \cdots \tilde{X}_{n/B}]$, the learning rate
       scheduler $\eta_t$ for Oja's method, the learning rate scheduler $\alpha_t$ for our acceleration
       and the number of eigenvectors to compute $k$
Output: The $k$ most significant eigenvectors of $X$
1 Initialize each column of $W_1$ randomly from the unit sphere in $\mathbb{R}^d$;
2 for $t = 1, 2, \dots n/B$ do
3     $W_{t+1} = \text{Normalize}\left(W_t + \eta_t \tilde{X}_t \tilde{X}_t^T W_t / B\right)$;
4     $\tilde{W} = W_{t+1} + \alpha_t W_t W_t^T W_{t+1}$;
5     $W_{t+1} = \text{Orthonormalize}\left(\tilde{W}\right)$;
6 end
```



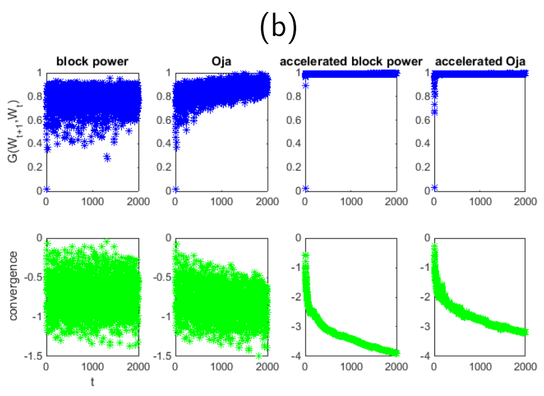


Figure 3.1: Performance in terms of the objective function $G\left(W_{t+1}, W_t\right)$ (Top) and the convergencece defined as $\log_{10}\left(1 - \frac{\|X^T W_t\|_F^2}{\|X^T V^*\|_F^2}\right)$ (Bottom) when computing the first eigenvector of the spiked covariance model according to [MCJ13]. (a) Performance of the accelerated mini-batch power method using decreasing, constant and increasing learning rates. (b) Performance of Oja's method and the mini-batch power method before and after applying our acceleration using increasing learning rates.

algorithm converges to a span of the eigenspace with basis vectors that significantly differ from the original eigenvectors. Furthermore, the performance of a specific streaming PCA method might vary significantly for different trials of exactly the same parameter settings due to the random initialization. In our study, we use the following function to evaluate convergence

$$\text{convergencece}\left(V^*, W_t\right) = \log_{10}\left(1 - \frac{\|X^T W_t\|_F^2}{\|X^T V^*\|_F^2}\right)$$

where $\|.\|_F^2$ is the squared Frobenius norm and $V^* \in \mathbb{R}^{d \times k}$ are the optimal k eigenvectors computed using standard PCA. Section 2.3.6 provides a discussion on the advantages of this metric. Many of the following experiments are performed on datasets that are too large to compute the eigenspace using the standard PCA. In such cases, we will refer to the mean-

Table 3.1: Summary of streaming PCA methods in terms of complexity and parameter dependence.

Method	FLOPs	space	mini-batch	online	parameter-sensitivity
Oja [OJE83; Oja82]	$\mathcal{O}(kd)$	$\mathcal{O}(kd)$	✓	✓	high
mini-batch power [MCJ13]	$\mathcal{O}(kd)$	$\mathcal{O}(kd)$	✓	✗	parameter-free
CCIPCA [WZH03]	$\mathcal{O}(kd)$	$\mathcal{O}(kd)$	✓	✓	parameter-free
IPCA [Aro+12]	$\mathcal{O}(k^2d)$	$\mathcal{O}(kd)$	✗	✓	parameter-free
SM [PSC18]	$\mathcal{O}(k^3+kd)$	$\mathcal{O}(kd)$	✓	✓	low
accelerated SGA (ours)	$\mathcal{O}(kd)$	$\mathcal{O}(kd)$	✓	✗	low

squared-error to assess convergence rates. Whenever data size permits, we run ten trials and report the average performance of each method.

3.4.1 Complexity and Parameter Dependence

The first important factors to compare are the space and time complexities and the parameter dependence of each method. Table 3.1 summarizes the space and time complexity per sample per update of each method with their parameter dependence. We did not consider the orthogonalization process in the complexity analysis of SGA methods since one does not need to perform it for each update. The *online* column indicates the methods that process one sample per update. One can note that CCIPCA enjoys the cheapest space and computational cost in addition to being parameter-free and generalizable to work in the mini-batch mode. IPCA is rather more expensive in terms of computational cost by a factor of k due to the EVD operation involved for the reduced-order matrix Q_n. As we discussed earlier, Oja's method, despite its very well established theoretical analysis and low computational cost, is not practical because of its performance sensitivity when using different initial learning rates. While our algorithm is still parametric, it is capable of finding the optimal solution using the stochastic scheduling strategies that we proposed. We show in all the experiments in the following sections that despite using fixed parameter configurations for the schedulers, our acceleration is still able to achieve the best convergence rates.

Table 3.2: Convergence of each method after a single data pass to the first five eigenvectors of the spiked covariance model generated with $k = 10$.

Method	fully online	B=10	B=100
Oja $\eta_t = \frac{1}{t}$	-0.87 ± 0.1	-0.83 ± 0.065	-1.47 ± 0.08
Oja $\eta_t = \frac{10}{t}$	-0.86 ± 0.05	-0.83 ± 0.065	-1.47 ± 0.08
Oja $\eta_t = \frac{100}{t}$	-0.82 ± 0.09	-0.83 ± 0.065	-1.47 ± 0.08
mini-batch (block) power	-	-0.84 ± 0.08	-1.46 ± 0.1
CCIPCA	-1.57 ± 0.14	-1.63 ± 0.17	-1.48 ± 0.1
IPCA	-1.53 ± 0.22	-	-
SM	-1.5 ± 0.14	-1.56 ± 0.13	-1.39 ± 0.1
accelerated Oja $\eta_t = \frac{100}{t}$ $\alpha_t = \frac{t}{1+z_t}$	-	-1.67 ± 0.15	-1.5 ± 0.1
accelerated Oja $\eta_t = \frac{100}{t}$ $\alpha_t = \frac{t}{1+1000\times z_t/t}$	-	$\mathbf{-1.81 \pm 0.2}$	$\mathbf{-2.41 \pm 0.2}$
accelerated block power $\alpha_t = \frac{t}{1+z_t}$	-	-1.68 ± 0.18	-1.5 ± 0.1
accelerated block power $\alpha_t = \frac{t}{1+1000\times z_t/t}$	-	$\mathbf{-1.88 \pm 0.2}$	$\mathbf{-2.4 \pm 0.17}$

Table 3.3: Run time (in seconds) of SM, IPCA and our acceleration after single pass on the spiked covariance model with $d = 1,000$ and $n = 10,000$.

	SM	IPCA	accelerated SGA $B = 10$	accelerated SGA $B = 100$
$k = 5$	0.93	1.33	0.75	**0.19**
$k = 10$	1.65	2.46	1.69	**0.4**

3.4.2 Convergence Analysis on The Spiked Covariance Model

We test each method on synthetic datasets generated using the spiked covariance model based on [MCJ13] where each time-step is drawn from the following generative model

$$x_i = Az_i + \sigma N_i,$$

where $A \in [-1, 1]^{d \times k}$ is a fixed matrix, $z_i \in \mathbb{R}^k$ is a random weight vector based on standard normal distribution and $\sigma N_i \in \mathbb{R}^d$ is a Gaussian noise vector of standard deviation σ. The task is to restore the component matrix A (up to unitary transformations) from the noisy samples x_i. We test the streaming techniques for the following settings $k = 10$, $d = 1000$, $n = 10,000$ and $\sigma = 1$. In such settings, the first k eigenvectors will be very close to each other in terms of their eigenvalues resulting in very small eigengaps. We would like to test these methods for recovering the first $q = 5$ eigenvectors instead of the the full 10 vectors. This is due to the fact that in practice, one does not always know the exact number of eigenvectors embedded in the model. Rather, such a choice is highly motivated by the computational capabilities. For the mini-batch methods, we set the mini-batch size to $B = 10$ and $B = 100$ in order to study convergence when using small mini-batch sizes. We run our acceleration using the two strategies proposed. All methods were initialized using the same random vectors from a unit sphere.

Table 3.2 shows the convergence results of each method. One can note that applying our acceleration using the second scheduling strategy significantly enhances the convergence of the mini-batch power iteration and Oja's method in addition to outperforming other state-of-the-art methods. Despite setting different learning rates for Oja's method, it leads to poor quality results. In general, increasing the mini-batch size for SGA methods results in better convergence unlike similarity matching and CCIPCA where there seems no clear relation between the block size and convergence. In terms of the fully online methods, CCIPCA converges better than IPCA, SM and Oja's method.

Table 3.3 reports the run time of Similarity Matching, Incremental PCA and our acceleration in order to have a better idea of the speedup gained when using the mini-batch mode. The algorithms were run on a machine with 3.7 GHz Intel Xeon CPU. Due to reduced order EVD involved, the IPCA takes the longest time compared to our acceleration technique and SM which is consistent with the computational complexity analysis shown in table 3.1. The accelerated mini-batch SGA when using a batch size of $B = 10$ has similar run time to SM which is due to the Gram-Schmidt orthogonalization involved. However, when using a larger batch size of $B = 100$ the run time was significantly reduced. Figure 3.2 shows the convergence rates based on a single trial where it can be noted that all mini-batch methods process data in a much shorter run-time compared to the fully-online streaming methods.

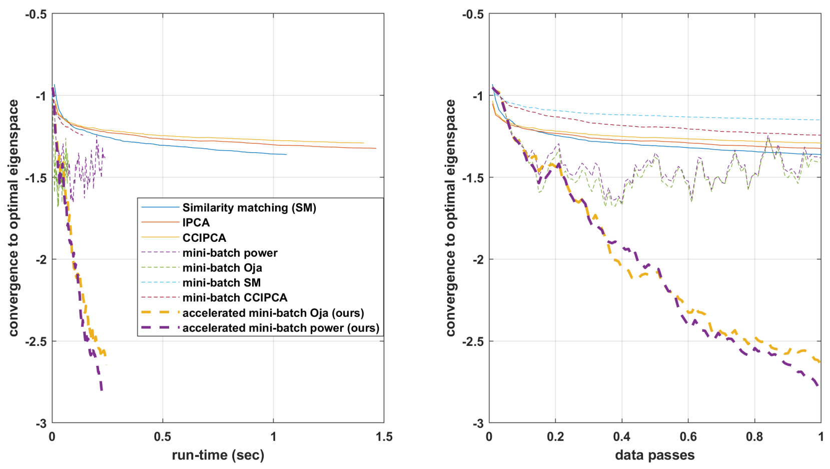


Figure 3.2: Convergence vs run-time and convergence vs data pass when computing the first five eigenvectors of the spiked covariance model. Note that applying our acceleration provides better convergence rates and shorter run-times.

Table 3.4: Convergence of each method after a single data pass to the first $k = 5$ eigenvectors of each dataset.

Method	MNIST	CIFAR-10	Fashion MNIST	Sign Language MNIST
mini-batch (block) power	-1.086 ± 0.0	-1.38 ± 0.007	-1.5 ± 0.006	-1.38 ± 0.003
CCIPCA	-2.22 ± 0.49	-3.2 ± 0.23	-3.55 ± 0.23	-2.9 ± 0.09
mini-batch CCIPCA	-2.45 ± 0.58	-2.57 ± 0.2	-3.49 ± 0.69	-2.69 ± 0.33
IPCA	-2.64 ± 0.003	-2.09 ± 0.0	-3.2 ± 0.0	-2.91 ± 0.0001
SM	-3.15 ± 0.35	-2.9 ± 0.37	-2.68 ± 0.34	-2.53 ± 0.55
mini-batch SM	-2.16 ± 0.6	-2.55 ± 0.51	-3.19 ± 0.61	-2.57 ± 0.35
accelerated Oja (1st strategy) $\eta_t = \frac{100}{t}$ $\alpha_t = \frac{t}{1+z_t}$	-2.32 ± 0.48	-2.54 ± 0.33	-3.39 ± 0.71	-2.8 ± 0.34
accelerated Oja (2nd strategy) $\eta_t = \frac{100}{t}$ $\alpha_t = \frac{t}{1+1000\times z_t/t}$	$\mathbf{-3.7 \pm 0.16}$	$\mathbf{-4.03 \pm 0.4}$	$\mathbf{-3.87 \pm 0.22}$	$\mathbf{-3.33 \pm 0.35}$
accelerated block power (1st strategy) $\alpha_t = \frac{t}{1+z_t}$	-2.6 ± 0.6	-2.69 ± 0.33	-3.37 ± 0.67	-2.83 ± 0.36
accelerated block power (2nd strategy) $\alpha_t = \frac{t}{1+1000\times z_t/t}$	$\mathbf{-3.88 \pm 0.07}$	$\mathbf{-4.01 \pm 0.4}$	$\mathbf{-3.89 \pm 0.21}$	$\mathbf{-3.34 \pm 0.34}$

3.4.3 Convergence Analysis on Benchmark Datasets

In order to have a better idea of the convergence using real-world examples, we analyze the convergence of each method on two well-known benchmark datasets, namely, MNIST [LeC98] and CIFAR-10 [KH+09]. The MNIST dataset consists of 60,000 grayscale images of hand-written digits each of resolution 28×28. The CIFAR-10 contains 50,000 RGB images of resolution 32×32. Both datasets are amongst the most highly regarded benchmarks in the machine learning community. We also study the convergence on Fashion MNIST [XRV17] and Sign Gesture MNIST [Kag] which are specially designed for benchmarking machine learning techniques besides the original MNIST dataset.

Table 3.5: Convergence of each method after a single data pass to the first eigenvectors of the noisy MNIST.

Method	k=1	k=5
mini-batch (block) power	-0.24 ± 0.2	-0.22 ± 0.011
CCIPCA	-1.9 ± 0.46	-2.04 ± 0.3
mini-batch CCIPCA	-2.2 ± 0.41	-1.89 ± 0.3
IPCA	-2 ± 0.5	-2.02 ± 0.25
SM	-2.35 ± 0.75	-2.16 ± 0.33
mini-batch SM	-1.96 ± 0.82	-1.83 ± 0.29
accelerated Oja (1st strategy)	-2.1 ± 0.49	-1.85 ± 0.26
accelerated Oja (2st strategy)	**-2.87 ± 0.07**	**-2.6 ± 0.1**
accelerated block power (1st strategy)	-2.42 ± 0.5	-2.04 ± 0.3
accelerated block power (2st strategy)	**-2.86 ± 0.04**	**-2.66 ± 0.17**

For the mini-batch (block) methods, we set the batch size to $B = 100$. As we mentioned earlier, we run 10 trials for each method and report the mean and standard deviation. Table 3.4 reports the convergence statistics for each method when computing the first $k = 5$ eigenvectors of each dataset. It is evident that applying our acceleration using the second scheduling strategy achieves the best convergence for all datasets. On the other hand, the mini-batch power without applying our acceleration provides the worst results due to the relatively small mini-batch size.

We also examine the convergence of streaming approaches in noisy conditions by applying a Gaussian noise on the MNIST dataset and then recovering the original eigenvectors using each technique. Figure 3.3 shows several samples before and after adding Gaussian noise of standard deviation $\sigma = 1$. Interestingly, we noticed that in such noisy settings, the standard PCA perfectly recovers the optimal $k = 5$ eigenvectors. Table 3.5 depicts convergence results when recovering the first $k = 1$ and $k = 5$ eigenvectors. The best performance is achieved when using the second scheduling strategy of our acceleration. Figure 3.4 shows the the recovery of the first three eigenvectors of the noisy MNIST using each method. One can note that our acceleration and IPCA are able to recover the first three eigenvectors accurately. On the flip-side, similarity matching and CCIPCA converges to a span of the eigenspace resulting in basis vectors that significantly differ from the original eigenvectors.

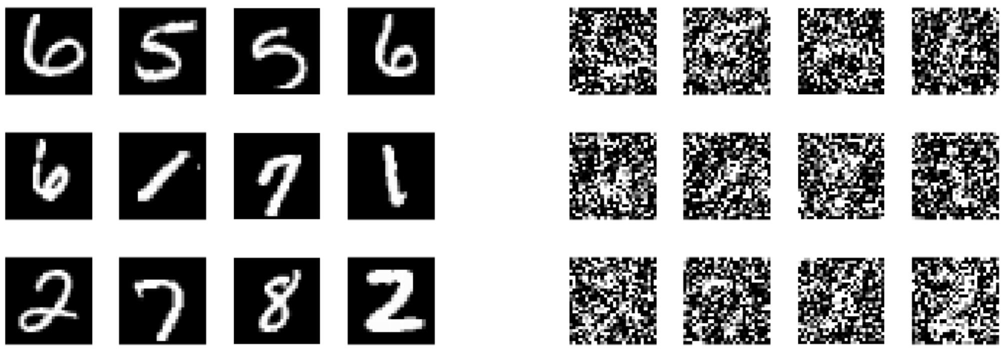


Figure 3.3: Several samples from MNIST data before and after adding Gaussian noise.

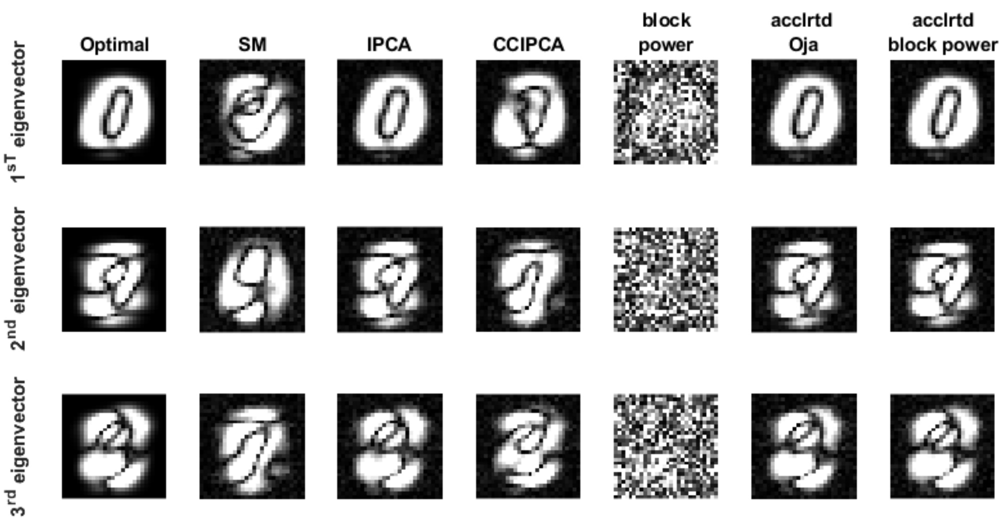


Figure 3.4: Recovery of the first three eigenvectors of the noisy MNIST using each method. Note that the eigenvectors derived using SM, CCIPCA and block power do not resemble the optimal eigenspace.

3.4.4 Convergence Analysis on Larger Datasets

Up to this stage, all the experiments were conducted on datasets that are small enough to acquire the optimal solutions using the standard PCA. In this section, we study the performance of streaming PCA methods on larger-scale datasets where applying the standard PCA is not feasible. Moreover, many of the datasets that will be studied cannot fit into the RAM space of a typical modern machine. Table 3.6 includes a summary of the datasets investigated in this section. The Labeled Faces in the Wild (LFW) dataset is one of the most well-known face datasets consisting of over 13,000 face images of 1,680 subjects (most subjects are famous politicians). The CelebA face dataset is similar to LFW but much more abundant in scale with over 200,000 images of 10,177 celebrities. Finding the eigenvectors (eigenfaces) of such datasets is of value. To the best of our knowledge, computing colored eigenfaces for large-scale face datasets is not widely discussed in the literature due to the significant extra dimensionality involved when considering all color channels. The LSUN bedroom dataset (part of the Large-scale Scene Understanding challenge) is the newest, but most abundant in scale.

Since we cannot acquire the optimal solution using the standard approach, the convergence will be measured in terms of the average Mean-Squared-Error (MSE) between original samples and reconstructed ones after computing the first five significant eigenvectors using each technique. We compute the first five eigenvectors using each technique and compare the results. For the mini-batch methods, the batch size is set to $B = 100$. Table 3.7 shows the MSE achieved by each method after a single data pass. One can note that in all experiments, our second acceleration strategy produces the lowest errors. In addition, it can be noted that extending

SM and CCIPCA to work in the mini-batch mode does not always result in better accuracy. Amongst all techniques, the mini-batch power results in the highest errors due to the relatively small size of mini-batches. Figures 3.5 and 3.6 show the first five significant eigenfaces of the LFW and CelebA datasets computed using each technique. The eigenvectros produced using IPCA, CCIPCA and our acceleration seem consistent. In contrast, SM is producing the eigenvectors in reversed order.

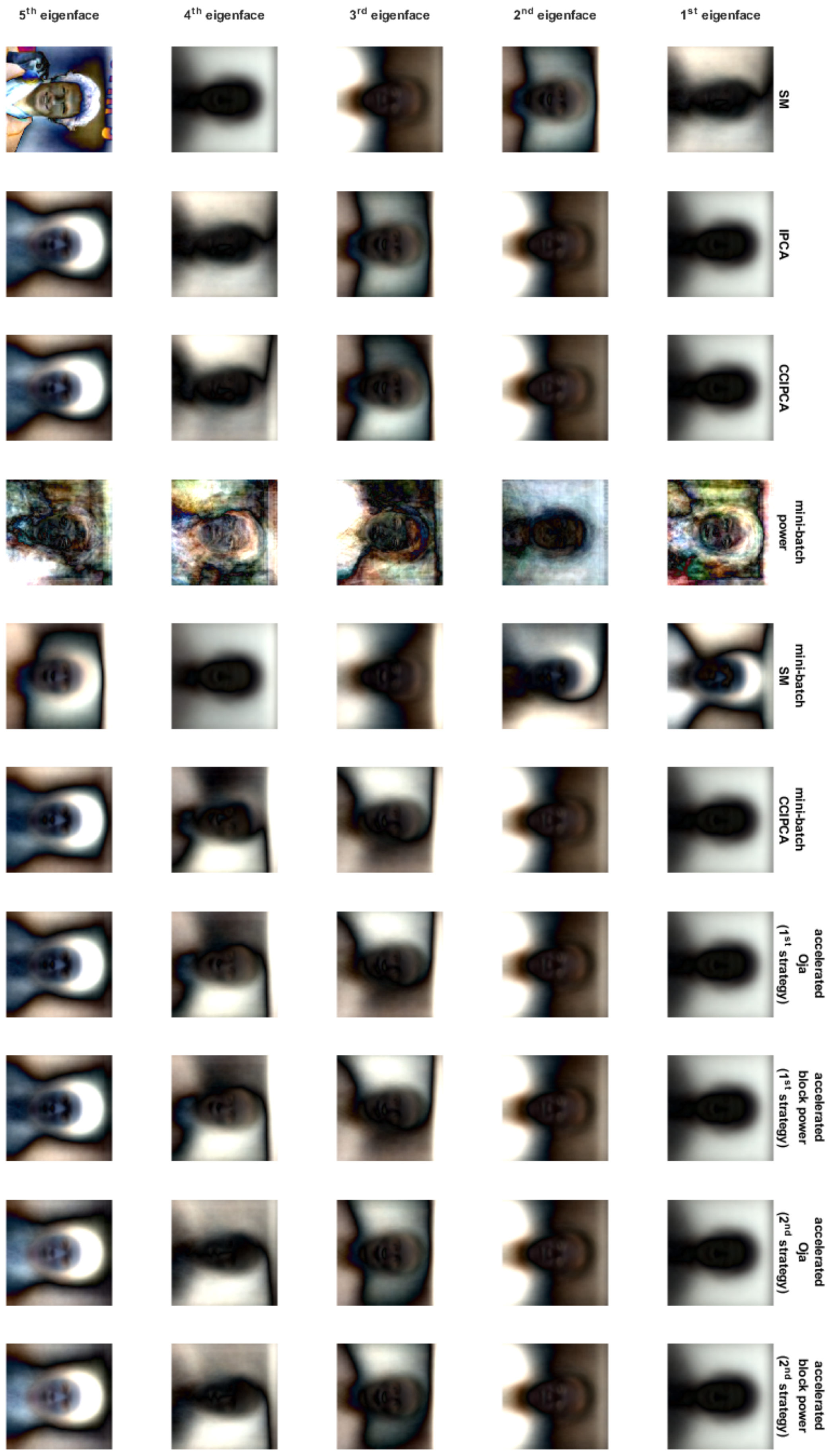
In order to get a better idea of the reconstruction of images using larger number of eigenfaces, we computed 200 eigenfaces using the accelerated mini-batch power method and then reconstruct the face samples. This gives a better MSE of 0.0091. Figure 3.7 compares different images from LFW data and their reconstructions when using 200 eigenfaces. It can be seen at a glance that the reconstructed images are rather blurry especially in the background areas. Also one can note that many faces are not well-aligned with their reconstructions. Only general facial features are preserved including basic facial expressions, rotation (to some extent), shape, ethnicity, hairstyle and coverings. One other thing to mention is that side faces are reconstructed in their nearest frontal pose and that occluding objects are filtered out. When reconstructing noisy samples, eigenfaces seem to be robust to noise as can be seen in figure 3.8. One of the problems in using eigenvectors as a holistic representation is determining the appropriate number of basis vectors to form the eigenspace. Such a decision is hard to make especially if the distribution of the streaming data is unknown. Rather, such a choice is based on the scree plot of the full set of eigenvalues [ZG06] which is hard to compute in streaming applications.

Table 3.6: Summary of the large-scale datasets used in the experiments.

	LFW faces	**CelebA faces**	**LSUN bedroom**
no. of samples	13,233	202,599	3,033,042
sample type	250×250 RGB images	178×218 RGB images	256×256 RGB images
Reference	[Hua+07; Lea14; HJL07; Hua+12]	[Liu+15]	[Yu+15]

Table 3.7: MSE achieved by each method when computing the first five eigenvectors after a single pass on the corresponding dataset.

Method	**LFW faces**	**CelebA faces**	**LSUN bedroom**
mini-batch (block) power	0.042395	0.038	0.040677
CCIPCA	0.040468	0.0363	0.038781
mini-batch CCIPCA	0.040478	0.0366	0.038768
IPCA	0.040462	0.0368	0.038796
SM	0.041548	0.0368	0.038777
mini-batch SM	0.040488	0.0366	0.038771
accelerated Oja (1st strategy) $\eta_t = \frac{100}{t}$ $\alpha_t = \frac{t}{1+z_t}$	0.040488	0.0363	0.038766
accelerated Oja (2nd strategy) $\eta_t = \frac{100}{t}$ $\alpha_t = \frac{t}{1+1000\times z_t/t}$	**0.040412**	**0.0362**	**0.038758**
accelerated block power (1st strategy) $\alpha_t = \frac{t}{1+z_t}$	0.040487	0.0363	0.038768
accelerated block power (2nd strategy) $\alpha_t = \frac{t}{1+1000\times z_t/t}$	**0.040408**	**0.0362**	**0.038758**



1st eigenface
2nd eigenface
3rd eigenface
4th eigenface
5th eigenface
SM
IPCA
CCIPCA
mini-batch power
mini-batch SM
mini-batch CCIPCA
accelerated Oja (1st strategy)
accelerated block power (1st strategy)
accelerated Oja (2nd strategy)
accelerated block power (2nd strategy)

Figure 3.5: Recovery of the first five eigenfaces of the LFW dataset using each method. Note that the generated eigenfaces seem consistent, excluding the block power and SM methods.

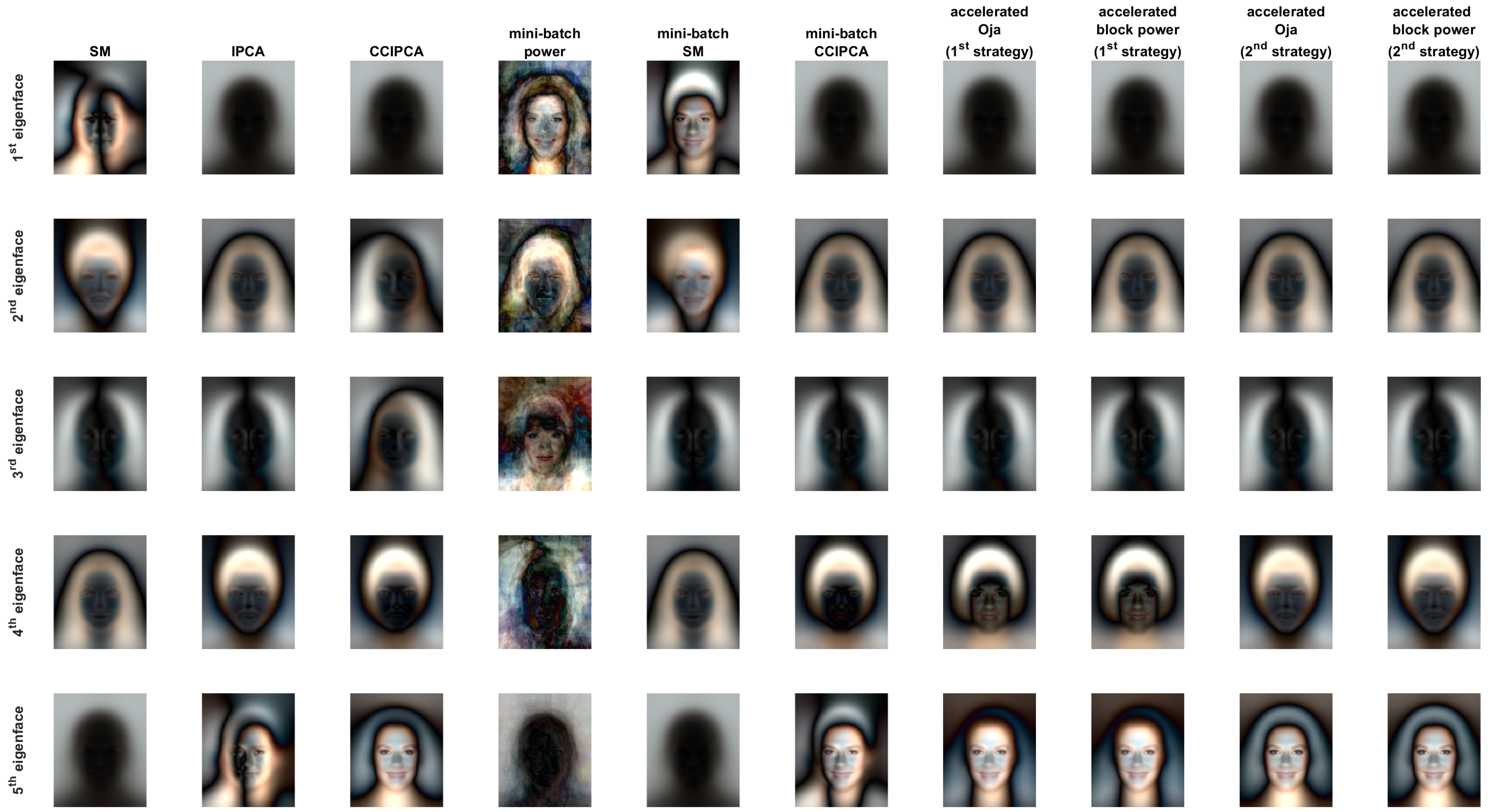

SM
IPCA
CCIPCA
mini-batch power
mini-batch SM
mini-batch CCIPCA
accelerated Oja (1st strategy)
accelerated block power (1st strategy)
accelerated Oja (2nd strategy)
accelerated block power (2nd strategy)
1st eigenface
2nd eigenface
3rd eigenface
4th eigenface
5th eigenface

Figure 3.6: Recovery of the first five eigenfaces of the CelebA dataset using each method. Note that the generated eigenfaces seem consistent, excluding the block power and SM methods.

3.4.5 Convergence Comparison with The Power Method Under Low Eigengap Settings

In this section, we study the performance of streaming methods when running multiple data passes and then compare the results with the batch power method which is one of the most popular reduced-complexity PCA solvers [GV12]. As discussed in section 2.2.1 in the background chapter, the convergence of this method to the i^{th} eigenvector depends on the eigengap $\Delta_i = \lambda_i - \lambda_{i+1}$. While enjoying linear convergence rates, the convergence of this method may be extremely slow when the eigenvectors are very close in their eigenvalues.

Figure 3.9 shows the results. It can be clearly noted that when initializing the power iteration randomly from the unit sphere, the method gives very slow convergence due the the small eigengap problem. On the other hand, when initializing the power method using our technique after running a single data pass, the convergence results of the power iteration were substantially improved. In general, one can note that streaming PCA approaches converge very fast, but in a sublinear manner. In other words, after few data passes, each algorithm reaches a stable point where no better improvements can be made. What we exactly did here is that we used the advantage of fast (but rather sublinear) convergence of streaming PCA to initialize the power method in order to enhance the convergence results. One last point to highlight is that even under low eigengap settings, applying our acceleration leads to excellent convergence after a single data pass in comparison to other state-of-the-art streaming approaches as well as the batch power method.

3.5 Limitations

The streaming PCA model, while significantly reducing the computational cost, suffers key limitations related to its scalability. Particularly, due to Gram-Schmidt orthogonalization processes involved, the studied streaming PCA approaches may have a dominant computational cost of $\mathcal{O}(k^2d)$ FLOPs per update. While such orthogonalization should not be performed after each update, this is not the case when considering time-critical applications that require monitoring the eigenvectors after each update, as will be visited later in Chapter 6. This causes a scalability problem when the number of eigenvectors is very large $k \gg 1$. In fact, considering in this case that the overall complexity of most streaming PCA methods is $\mathcal{O}(k^2nd)$, one can note that when $k^2 = \mathcal{O}(d)$, the computational complexity becomes $\mathcal{O}(nd^2)$ which is equivalent to the complexity of standard PCA. Furthermore, streaming PCA methods are highly sequential with linear computational dependence on data dimensionality d which limits the use of parallel programming resources such as GPUs and makes designing an efficient hardware accelerated implementation not an easy task. This is even more problematic when considering massive datasets in which fitting a single sample into the memory space is not possible.

3.6 Conclusions

In this chapter, we proposed an acceleration scheme for SGA-based streaming PCA methods. Particularly, our approach aims to accelerate such algorithms in the mini-batch mode, where each update step processes multiple samples of the dataset. Experiments on the spiked covariance model, MNIST, CIFAR-10, LFW, and other large-scale benchmark datasets showed that applying our scheme to accelerate Oja's method and the mini-batch power method using the second learning strategy leads to a significant improvement in the convergence rates in addition to outperforming other state-of-the-art streaming PCA algorithms. We also showed that in the case of low eigengap, our acceleration could be used to initialize the power method after running a single data pass which leads to a substantial improvement in the convergence results. In short, our acceleration scheme leads to significantly faster convergence rates and reduced computation run-time. While our scheme was shown to converge very well using stochastic strategies for scheduling the learning-rates, investigating other strategies to improve further the performance would be an interesting topic for further research.

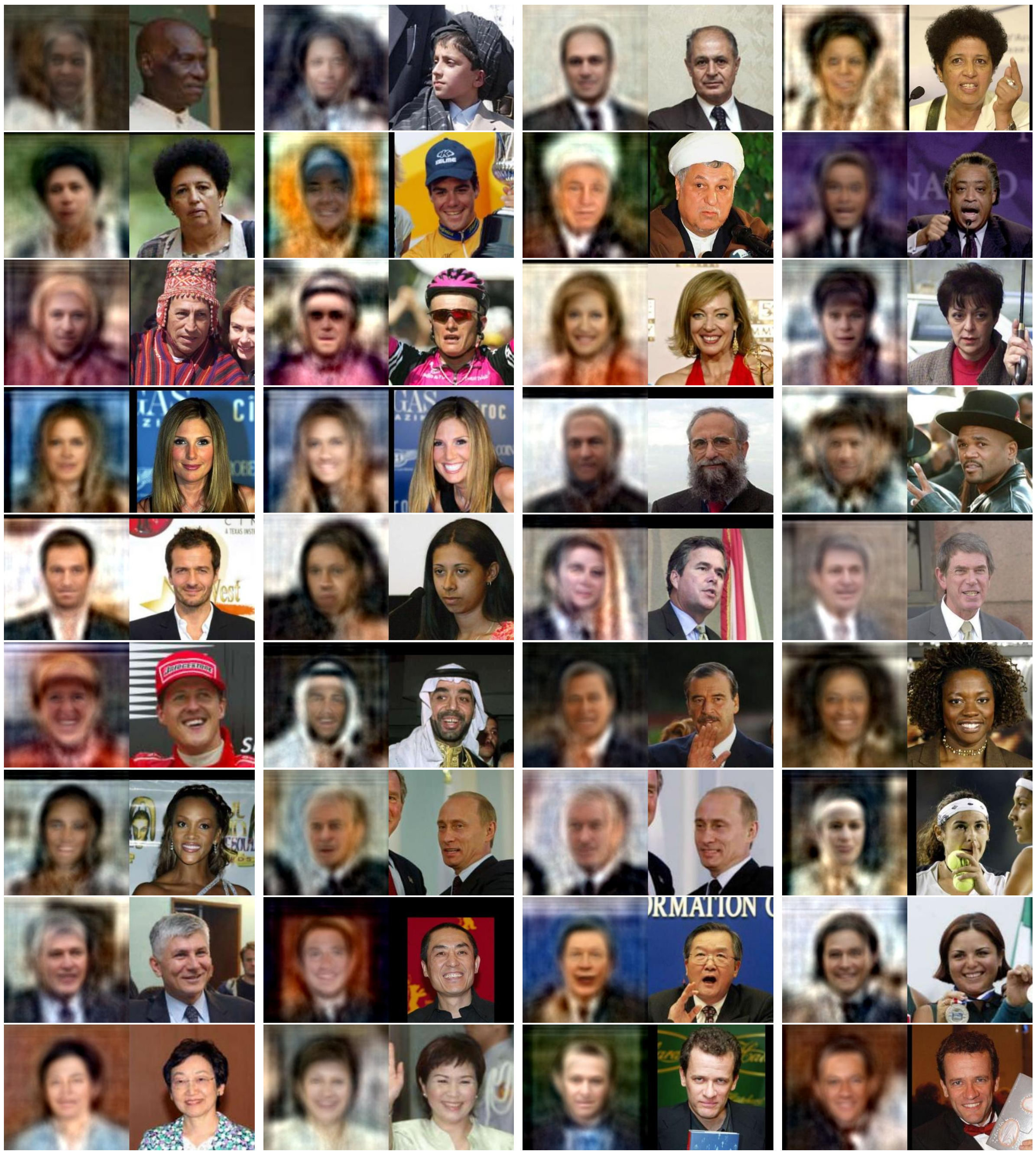


Figure 3.7: Various samples from Labeled Faces in the Wild reconstructed after computing 200 eigenfaces using accelerated block power. Note that the reconstructed images are very blurry.

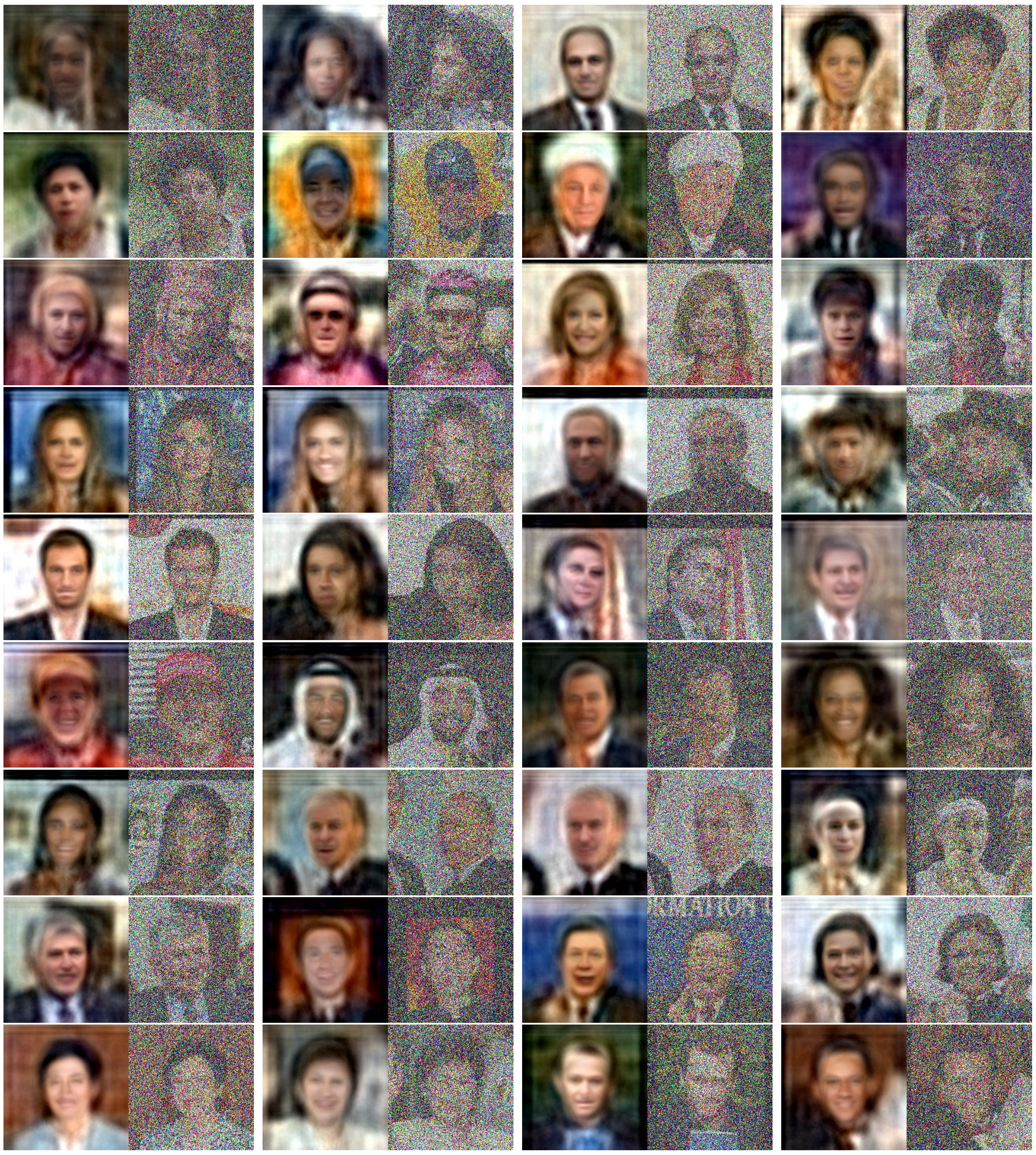


Figure 3.8: Noisy samples from Labeled Faces in the Wild reconstructed after computing 200 eigenfaces using accelerated block power. Note that the noise does not significantly affect the reconstruction quality.

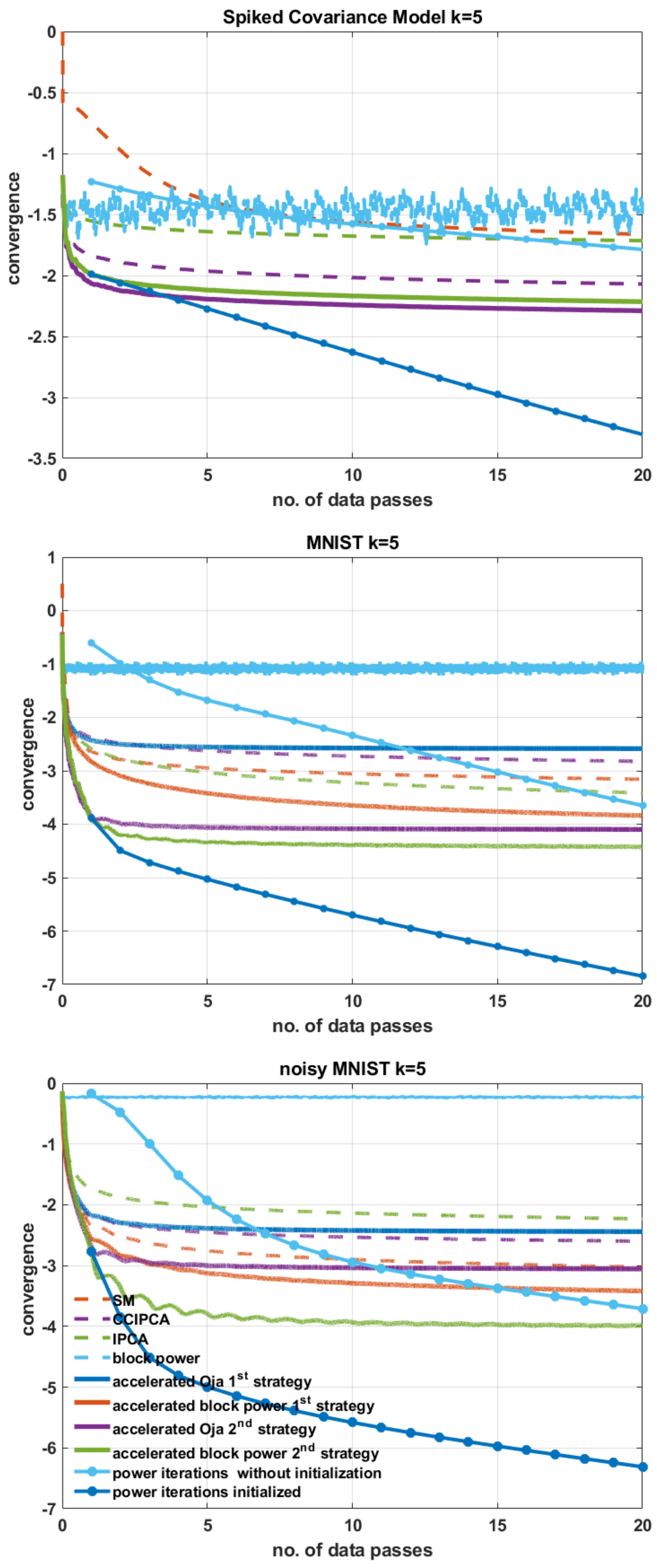


Figure 3.9: Comparison between the convergence of streaming methods and the batch power method when performing multiple data passes. Note that the convergence of the batch power method, when initialized using the eigenvectors estimation of the accelerated block power, is significantly better compared to the convergence when applying random initialization.

4 A Parallel PCA Model for High Dimensional Data

In the previous chapter, we discussed different streaming PCA methods for modelling large scale datasets. In all experiments, we applied such methods to derive a holistic representation for the studied datasets. One problem when applying streaming PCA methods in such contexts is the fact that these type of methods are highly sequential and not well-scalable when computing a large number of eigenvectors $k \gg 1$. We also show that applying streaming PCA in the mini-batch mode enhanced the run time and reduced the scalability problem to some extent. In this chapter, we propose a different methodology for modelling large-scale datasets. The model is simple and highly parallel overcoming the aforementioned bottlenecks.

4.1 Introduction and Model Description

Recall that the main task of PCA is to compute low-dimensional basis vectors which capture most of the variability of the input dataset $X \in \mathbb{R}^{d \times n}$. These basis vectors correspond to the k most significant eigenvectors $V \in \mathbb{R}^{d \times k}, k \ll \min(n, d)$ of the covariance matrix $C = \frac{1}{n-1} X_c X_c^T$ where X_c is the dataset after subtracting the sample mean. Such a holistic linear representation is optimal in terms of the mean-squared-error [Jol02]. However, finding such basis vectors requires $\mathcal{O}(nd \min(n, d))$ FLOPs of computation and $\mathcal{O}(\min(n, d)^2)$ memory space. Hence, analyzing large-scale datasets (of very large $\min(n, d)$) becomes computationally infeasible. We explored in the previous chapter one common type of solution, namely, streaming PCA which reduces the order of complexity from quadratic to linear. One problem that recurs in methods of this type, as discussed in the previous chapter, is the fact that they are not parallel and bear the assumption that the sample dimensionality d is addressable within the available machine memory space. In many cases, the dataset may be too large that even a single sample may not fit into a typical modern machine space.

In this chapter, we investigate a different type of solution. The idea is basically to subdivide each sample into smaller partitions where each partition constitutes a particular subset of attributes and then apply PCA to each partition separately. While this idea is simple, it has many major advantages over the traditional holistic approach. By subdividing samples into p partitions of equal sizes, one can note that the resulting p covariance matrices are smaller than the holistic one since $p.(d/p)^2 = d^2/p < d^2$ and the computational complexity becomes $\mathcal{O}(nd^2/p)$ instead of $\mathcal{O}(nd^2)$ (assuming $d < n$) reducing both space and computational cost. Moreover, since each partition is processed independently, the approach is highly parallel and the computation can be further reduced to $\mathcal{O}(n(d/p)^2)$. Having these advantages on hand, such a mode of computation raises two main questions. First of all, on what basis are attributes mapped to different partitions? We propose two different strategies for assigning attributes to different partitions which we refer to as Cell-based PCA (CPCA) and Band-based PCA (BPCA). We asses each strategy in terms of the average mean-squared-error and SSIM between reconstructed samples and their original counterparts in addition to reporting the run-times when applying CPU and GPU implementations. We find that using the proposed partitioning strategies significantly enhances reconstruction and dramatically reduces the run-time. The second question concerns the relation between the partitioned eigenvectors and the holistic PCA model. Experimental results indicate that when assigning attributes to each partition randomly, the combination of the resulting partial eigenvectors becomes analogous to the holistic solution. This suggests that even the baseline instance of the partitioning model may be a more practical alternative to the streaming PCA approach due to its parallel and scalable nature.

4.2 Related Work

Partitioning-based PCA has become an active research area in the last two decades specifically for applications in distributed data analysis, computer vision and computer graphics. Partitioning-based PCA algorithms can be categorized into two types: Sample-based partitioning and attribute-based partitioning. The sample-based partitioning is the most commonly used approach with profound use in domain applications, including subspace clustering and distributed systems. Such a family of approaches divides the data into smaller subsets of samples for two different goals depending on the type of application. In subspace clustering, the aim is to compute a set of subspaces where each subspace optimally fits a subset of samples based on their distribution [Vid11]. It has been widely employed for video and motion segmentation [VMP04; VF14], classification [EV13; EV09] and multi-variate volume visualization [Liu+14]. On the other hand, distributed PCA addresses the problem of analyzing data partitioned across multiple distributed servers. It can be used either as sample-based partitioning where each server possesses $n_i < n$ samples of high dimension or attribute-based partitioning where each server streams $d_i < d$ channels of partial attributes to a global coordinator. The first scenario is typical when considering very high dimensional data distributed in star-topology networks where each machine performs a local SVD on their samples, followed by global processing using the center coordinator. Such a problem has also been explored in the machine learning community [Qu+02; KVW14; GSS17]. A more challenging scenario is that when servers are connected in mesh-topology networks which was discussed by Wu et al. [Wu+18] and Fellus et al. [FPG14]. In this setting, the power method is applied in a way where after each local update, a globalization procedure is performed using the Gossip communication protocol [Tsi84; Boy+06; Dim+10]. One can note that in the sample-based partitioning in distributed PCA, the general problem remains a batch type of learning. On the other hand, attribute-based distributed PCA is more commonly used in streaming applications, particularly in the field of multi-sensor signal processing. Attribute-based partitioning was also investigated for learning the appearance of 3D mesh models using eigen-textures [NSI99] where PCA is applied to each texture segment of a rotating object. Diamantaras and Kung [DK93] used PCA basis functions instead of DCT for encoding Intra-frame blocks in the MPEG protocol. They found that the reconstruction quality achieved by PCA was significantly better in comparison to DCT. Ye et al. proposed a PCA-based dimensionality reduction method that preserves the spatial structure in the reduced space [YJL04]. However, the proposed method works only in offline mode and is not suitable for streaming applications. In this chapter, we will study attribute-based partitioning PCA in a more general context where we will explore different partitioning models and compare such models with the standard holistic approach in terms of computation and reconstruction quality.

4.3 Cell-based PCA (CPCA)

In this section, we address the first strategy for assigning attributes which we refer to as Cell-based PCA (CPCA). The technique is inspired by the JPEG compression standard [Wal92] where images are subdivided into small blocks (cells) of size 8×8. Each block is then projected onto the 2D Discrete Cosine Transform (DCT) basis functions. These basis functions form a global representation for any 8×8 graye-scale images by defining spatial frequencies in the 2D space as discussed in the background chapter (section 2.5.2). Each block is represented in terms of its projection values on these 64 basis functions. In order to achieve compression, only a few projection values are chosen for representation, those corresponding to lower frequency basis functions. The compression step is done via the use of a quantization-matrix (Q-matrix). The Q-matrix is an 8×8 matrix consisting of divisors for the corresponding projection values. Each projection value is divided by its corresponding factor and then rounded in order to achieve quantization and compression by neglecting zero rounded resultants. Hence, the higher the divisor value the more chance the projection value will be neglected. There is no trivial way of defining the Q-matrix. However, the Independent JPEG Group (IJG) tables are the most commonly used Q-matrices [Kor08]. One can note that the main drawback in such a representation is the difficulty of prioritizing the low-frequency basis functions in the spatial domain. This becomes even more problematic when dealing with larger block sizes or three-dimensional volumetric data where the data are subdivided into smaller 3D blocks.

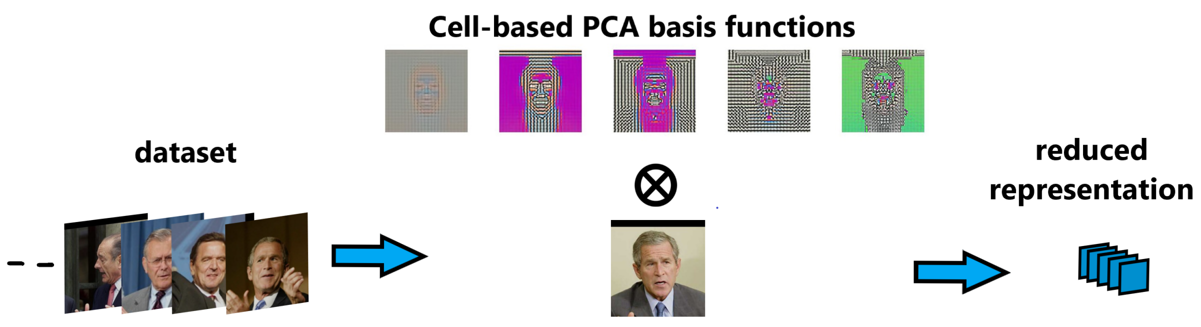


Figure 4.1: Representing face images using cell eigenfaces.

Unlike JPEG compression, CPCA is a data-driven representation and compression approach. In this case, basis functions of each individual block are computed by applying PCA to pixels within the region occupied by the corresponding cell. This brings the advantage of finding optimal basis functions of each block appropriately ordered based on their significance leading to minimal reconstruction errors [Jol02] when using a fixed number of basis functions. The downside of this method is that each block will have its own basis functions (eigencells) based on the distribution of the input dataset. In our implementation, we include all color channels for computing eigencells. This is an obvious choice since one would expect the distribution of colors in a single block to be consistent. Computing cell eigenvectors for larger cell sizes may not be computable using the standard PCA. For such scenarios, streaming PCA can be applied instead. Figure 4.1 illustrates the CPCA representation steps when considering a dataset of face images.

4.3.1 A First Example

As a first test, we applied CPCA on LFW and CelebA face datasets where the cell size was set to 10×10 RGB pixels. Hence, each cell will contain at most 300 attributes (considering edge-cells). One can note that computing the optimal eigenvectors using standard PCA for such a cell size is doable. The eigenvectors of each cell were obtained by computing its covariance matrix and then applying eigenvalue decomposition (EVD) to the resulting covariance matrix. Note that EVD is better than SVD in this case due to the large number of samples in the studied datasets. We used only 20 eigencells (eigenvectors per cell) for encoding and reconstructing images, resulting in a 15:1 compression ratio. In this thesis, we use the term *compression ratio* to describe the reduction in scalar values used for data representation. This terminology was used similarly by many studies [NSI99; YJL04].

Table 4.1 summarizes the results in terms of the average MSE and the PCA computation run-time per cell for each dataset. It also compares the results with holistic PCA when computing 1,000 eigenvectors using our acceleration scheme presented in Chapter 3. The reported run-times are based on a machine with a 2.6 GHz Intel Core 6700HQ CPU. Recall that the LFW dataset contains 13,233 samples of $250 \times 250 \times 3$ pixels and CelebA consists of 202,599 images of $178 \times 218 \times 3$ pixels. The reported run-times take into consideration the incremental computation of the cell covariance matrices where increments were performed using mini-batches of size 100. One can see at a glance, that computing cell-based PCA provides a significantly better reconstruction quality and significantly shorter computation time compared to the holistic eigenvectors. The MSE when reconstructing CelebA samples is higher than the one of LFW which may be due to the much larger number of samples in the CelebA dataset. Another advantage of using CPCA is that one can easily specify the compression ratio on-demand by changing the number of cell eigenvectors since applying standard PCA gives the full set of 300 basis vectors per cell. In addition, reconstruction using eigencells is parallel and can be achieved progressively due to the orderliness in such a representation. Figure 4.3 compares the reconstruction of a face image from LFW dataset when using only 20 cell eigenfaces with the reconstruction achieved using the first 1,000 holistic eigenfaces. It is evident that CPCA achieves a much better reconstruction quality while maintaining a higher compression ratio of 15:1 compared to 13.2:1 when using 1,000 holistic eigenfaces. Figure 4.2 shows reconstructed samples from LFW dataset using the resulting 20 cell eigenfaces where the lower two samples were excluded from the training set used for computing the eigenspaces. We can see that even background areas are clear. However, such reconstruction may cause subtle block boundary artefacts. Such boundary artefacts can be reduced by overlapping the cells boundaries or by using a deblocking filter. Figure 4.4 shows 20 CPCA basis functions where each cell differs from others based on the training data spatial distribution. We can note that lower eigencells count for higher spatial frequency details.

4.3.2 Adaptive Cell-based PCA

A further improvement can be made by adaptively varying the number of eigenvectors per cell in order to achieve a more optimal tradeoff between compression and quality. The number of eigenvectors per cell is determined based on the total variability explained by the first $k < k_{max}$ eigencells. This can be expressed as follows:

$$\Theta = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{j=1}^{d_{cell}} \lambda_j} > T \vee k < k_{max} \tag{4.1}$$

Table 4.1: Average mean-squared-error and run-time per cell (in seconds) for batch CPCA when applied to LFW and CelebA face datasets.

	LFW		**CelebA**	
Method	Cell-based PCA	Holistic PCA	Cell-based PCA	Holistic PCA
No. of eigenimages	20	1,000	20	1,000
MSE	0.00041	0.0029	0.0009	0.003
average run-time	0.16 Sec	hours	1.92 Sec	hours

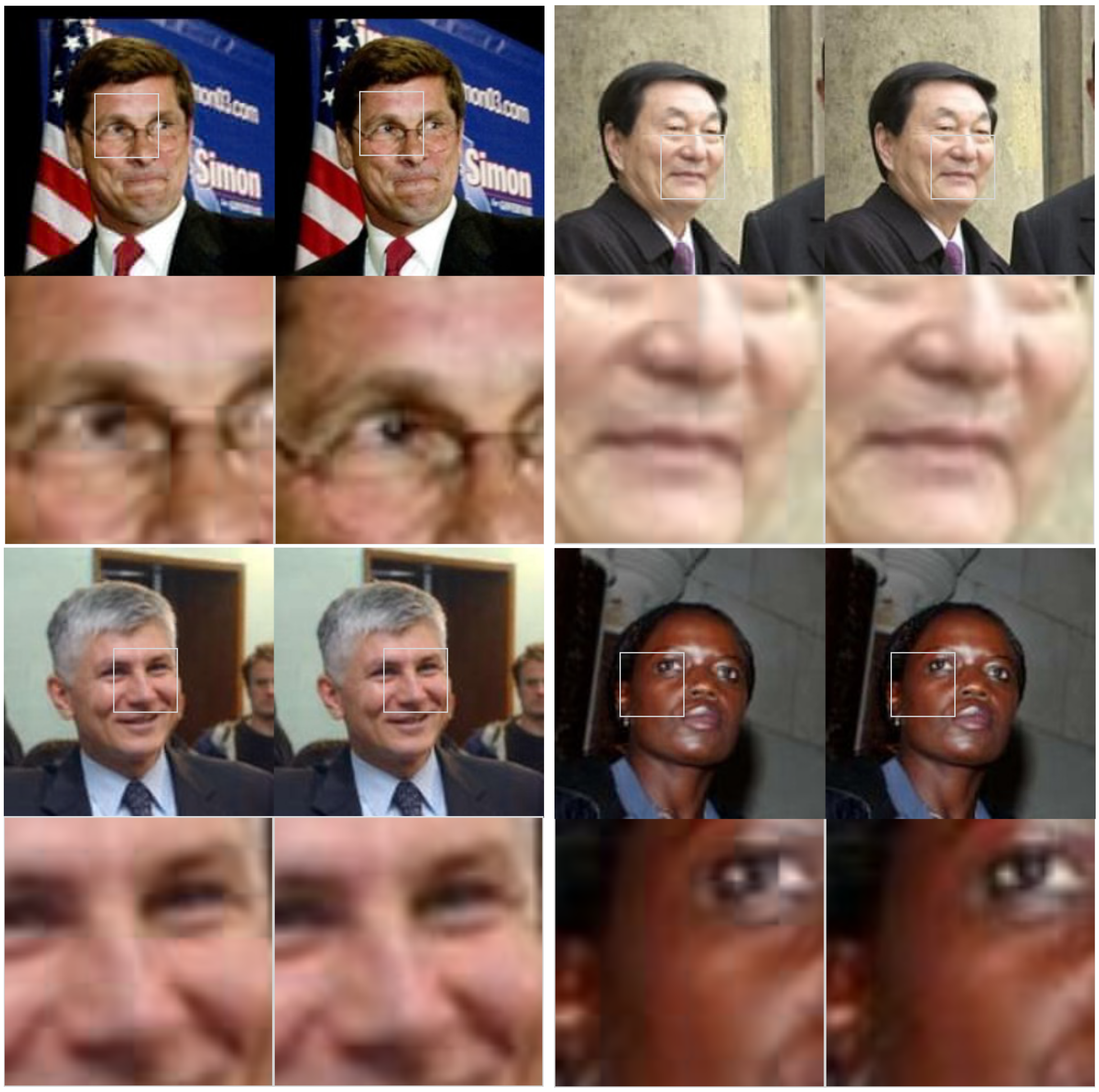


Figure 4.2: Comparing original samples from LFW dataset (right counterparts) with their reconstructions (left counterparts) when using 20 eigencells. Note the block artefacts in the zoomed areas.

where λ is the eigenvalue of the the corresponding eigenvector, k is the number of first significant eigenvectors, d_{cell} is the total number of attributes per cell and T is a threshold value, which affects the tradeoff between quality and average number of eigenvectors per cell (compression). Here, we bound the maximum number of eigenvectors per cell via k_{max} in order to guarantee a minimal compression of $d_{cell} : k_{max}$. We used the idea of applying CPCA with adaptive eigencells in the context of 3D volume visualization in [AD16b; AD16a; AD18b; AD18a].

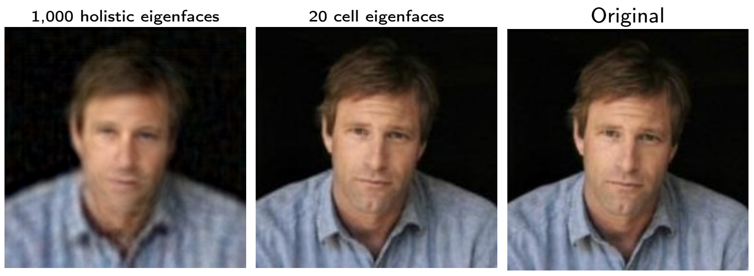


Figure 4.3: Reconstructed sample from LFW dataset using 1,000 holistic eigenfaces (with 13.2:1 compression ratio) compared to 20 cell eigenfaces (with 15:1 compression ratio). One can note the better reconstruction quality obtained when using CPCA despite the higher compression ratio.

4.4 Modelling non-Spatially Localized Data Using Band-based PCA (BPCA)

One of the main limitations of CPCA and DCT is that they work only for spatially structured datasets. In this section, we present an alternative partitioning technique that can be applied to more general types of datasets. The main idea of the approach is to assign attributes to each partition (band) based on the distribution of the feature values instead of their spatial locations. We discuss two different strategies for assigning attributes in the following subsections.

4.4.1 Mapping Attributes Based on Sample Mean

The main assumption JPEG and CPCA are based on is that neighboring attributes are usually very similar in terms of their pixel values. However, if the data on hand are not spatially organized, such an assumption is no longer valid. In this section, we make no spatial assumptions when assigning attributes to their corresponding bands. Rather, we group attributes that are close in terms of their mean values which can be described as follows

$$B = \{a \in x \mid l_B \leq \mathbb{E}(a) \leq u_B\}, \tag{4.2}$$

where a is an attribute value of x and l_B and u_B are lower and upper bounds defining the interval of the corresponding band. Typically, the difference between these bounds should be small. This can be done by subdividing the range of values in the sample mean into s sub-intervals. This will give a non-uniform number of attributes per interval. In order to limit the number of attributes per band, attributes belonging to the same interval are further sorted based on their mean values and then each L ordered attributes are mapped to a single band. The main problem is that computing the sample mean requires a pre-processing data pass, which is prohibitive in case of streaming data scenarios. One can solve this problem by estimating such statistics from a single mini-batch of the input data assuming samples are

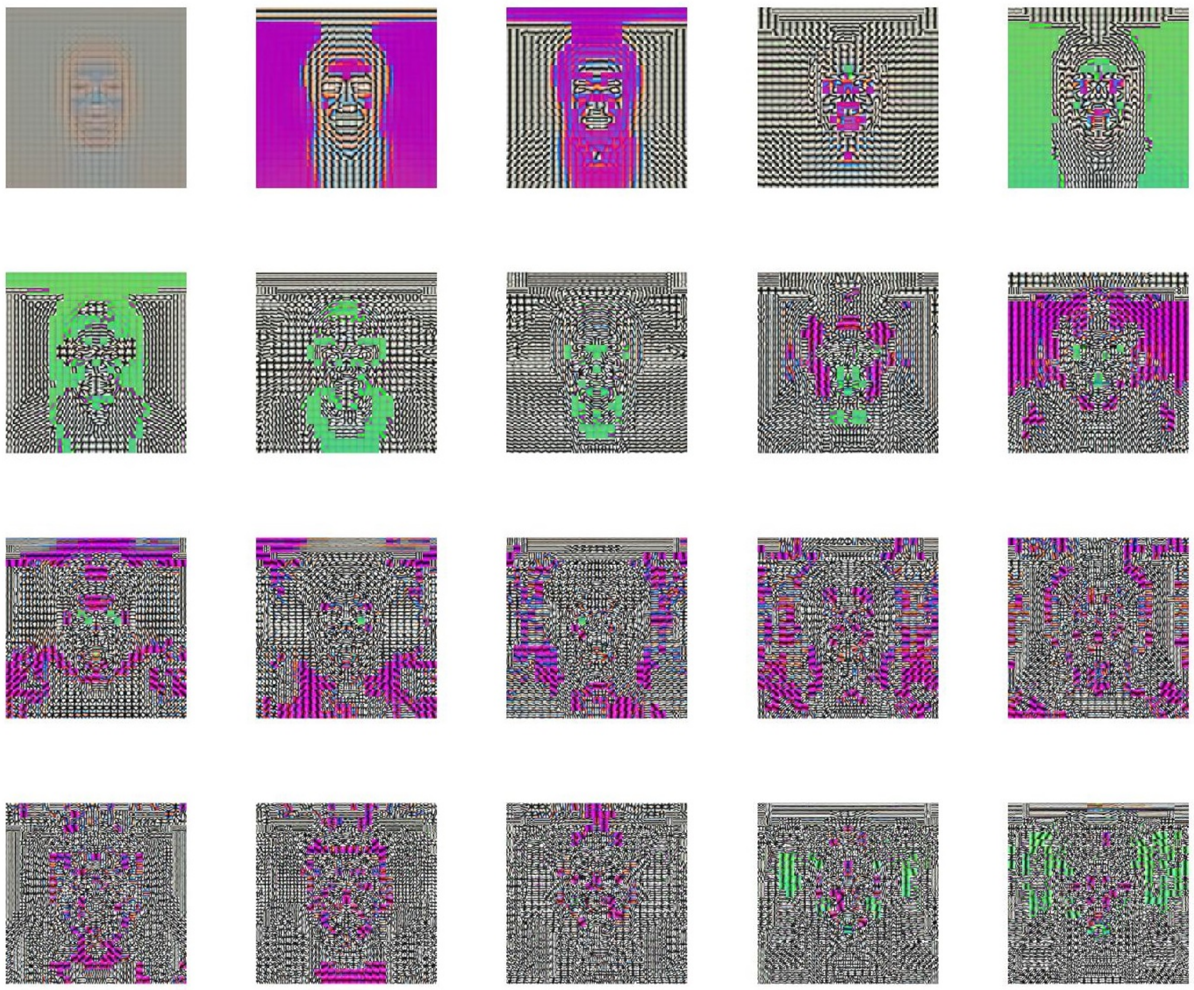


Figure 4.4: First 20 eigencells of the LFW dataset combined based on their spatial positions which we refer to as cell eigenvectors (cell eigenfaces in this case).

independent and identically distributed.

4.4.2 Mapping Attributes Based on Mean and Variance

Using only the sample mean as a basis for mapping attributes may neglect important aspects that attributes of the same band must share. Furthermore, using the mean by itself does not constitute the distribution of values that a particular attribute may have. It is well-known that PCA is most effective when samples obey a multivariate normal distribution [HL13]. Hence, we may assume that attributes of the same band are normally distributed and share similar parameter values in terms of mean and standard deviation. More technically, this can be expressed as follows

$$B = \left\{a \sim \mathcal{N}\left(\mu, \sigma^2\right) \in x \mid l_B^m \leq \mu \leq u_B^m \wedge l_B^s \leq \sigma \leq u_B^s\right\}, \tag{4.3}$$

where $\mu = \mathbb{E}(a)$ and $\sigma^2 = \text{Var}(a)$ are the mean and variance respectively of the attribute *a*. In order to achieve such conditions, we subdivide the range of values in the sample mean into *s* sub-intervals. After that, we sort samples based on the variance (instead of the mean) and then group each *L* ordered attributes into one band.

4.5 Results

We will evaluate the performance of the aforementioned techniques on two large-scale face datasets, namely, Labeled Faces in the Wild (LFW) [Hua+12] and CelebA [Liu+15]. While generalizable to more general types of datasets, our choice of such datasets is to build upon an application where the impact of the holistic PCA has a well-earned reputation. In addition, this allows for visual inspection of the reconstruction quality. It is worth mentioning that computing the holistic eigenspace for such datasets is infeasible due to the size of the dataset. However, we approximate the first eigenfaces using streaming PCA. In particular, we employed the accelerated mini-batch power using the 2^{nd} learning strategy where $\alpha_t = \frac{t}{1+1000\times z_t/t}$.

4.5.1 Reconstruction Quality for Different Cell Sizes

We first study the effect of varying the cell-size on the reconstruction quality when maintaining the same compression ratio of 15:1. Table 4.2 contains the average MSE and SSIM scores for different cell sizes compared to the reconstruction of 1,000 holistic eigenfaces. We can clearly note that the CPCA results are much better than the holistic scheme. This is well-reflected in figure 4.5 where cell boundary artefacts are reduced when increasing cell-size from 5×5 to 25×25. Figure 4.6 shows reconstructed samples from LFW and CelebA using different numbers of cell eigenfaces with a cell size of 25×25.

Table 4.2: CPCA vs. holistic PCA in terms of reconstruction quality.

	LFW				CelebA			
Cell-size	**$5\times 5 \times 3$**	**$10\times 10 \times 3$**	**$25\times 25 \times 3$**	**Holistic**	**$5\times 5 \times 3$**	**$10\times 10 \times 3$**	**$25\times 25 \times 3$**	**Holistic**
# of eigenfaces	5	20	125	1,000	5	20	125	1,000
MSE	0.0009	0.0004	**0.00027**	0.0029	0.0014	0.0009	**0.0007**	0.003
SSIM	0.8859	0.927	**0.9425**	0.71	0.8375	0.8768	**0.89**	0.72

4.5.2 CPU Vs. GPU Implementation

We compare the run-time of CPU and GPU implementations in MATLAB for different cell-sizes based on a machine with a 2.6 GHz Intel Core 6700HQ CPU and a GeForce GTX 960M GPU. We find that for small cell-sizes the CPU takes lower run-time than the GPU. On the other hand, for larger cell-sizes, the GPU implementation is faster. This is depicted in table 4.3 where the average run-times per cell are reported. As mentioned earlier, not only the EVD operations were counted for the run-time but also the incremental computation of the cell covariance matrices. Each cell covariance matrix was updated in a mini-batch manner using a mini-batch size of 100. We found that even in this case, the mini-batch is advantageous in terms of processing speed. The main downside when applying CPCA is the large memory space required for storing the cell covariance matrices. For instance, applying CPCA on the CelebA dataset using a cell size of 25×25 costs 4GB of memory space. Having that said, the size of the resulting p covariance matrices is still smaller than the holistic one since $p.(d/p)^2 = d^2/p < d^2$. Besides this, approximating the top 1,000 eigenvectors of both datasets using streaming PCA costs several hours of computation.

Table 4.3: CPU vs. GPU average run-times per cell in seconds.

	LFW			CelebA		
Cell-size	$5\times 5 \times 3$	$10\times 10 \times 3$	$25\times 25 \times 3$	$5\times 5 \times 3$	$10\times 10 \times 3$	$25\times 25 \times 3$
CPU run-time/cell	**0.0427**	**0.16**	7.87	**0.44**	1.92	63.3
GPU run-time/cell	0.3273	0.295	**3.19**	0.9	**1.2845**	**4.39**

4.5.3 Comparing BPCA Performance for Different Mapping Strategies

We now compare the different strategies for BPCA. For image datasets, due to the redundancy in the color channels, we assign pixels to different bands based on one color channel (in this section, we used the R channel, but a more general approach would be to use a grayscale transformation). Since we are addressing streaming and large-scale data in this study, we estimate the sample mean and variance using a subset of the dataset which we refer to as the *estimating subset*. We study the reconstruction quality for different estimating subset sizes and compare with performance when using the whole number of samples for computing mean and variance. As we found earlier in this section that increasing the cell size enhances the reconstruction results, we set the maximum attributes allowed per band L to 1,875 and S to 50. We report reconstruction quality in terms of MSE and SSIM when using 125 band eigenfaces. Due to the way our mapping strategy assigns attributes, the number of attributes per band may be smaller than L for many bands resulting in different compression ratios than the desired target (15:1). We also test BPCA when applying random mapping where attributes are assigned to different bands in a random manner. Table 4.4 compares reconstruction results between the baseline random-mapping model and the two proposed mapping strategies for different estimating subset sizes. We studied the reconstruction quality on the LFW dataset. Both proposed mapping techniques are much better in reconstruction than the baseline model. It is evident that mapping attributes using the mean and variance gives better results compared to the mean-based technique. In addition, increasing the estimating subset size enhances the results. However, the reconstruction results for CPCA are still better despite using lower compression ratio. Figure 4.7 shows many images reconstructed using different mapping strategies of band based PCA. Clearly, random-mapping BPCA results in poor reconstruction quality, whereas mean mapping produces some dithering artifacts. These dithering artifacts are enhanced when using the mean-variance mapping.

4.5.4 Analogy with the Holistic Eigenspace

Figure 4.8 presents a comparison between holistic eigenfaces and combined band eigenvectors (band eigenfaces) resulting from random-mapping BPCA. Interestingly, the eigenfaces resulting from random-mapping BPCA show a high resemblance to the holistic ones. This is also depicted in figure 4.9, where the explained variance curves of each partitioning strategy are presented. Both CPCA and mean-variance BPCA cover much more variations compared to random-mapping BPCA and holistic PCA. This suggests that the baseline of BPCA, achieved when randomly mapping attributes to different subsets, produces an eigenspace that is analogous to the holistic solution. The impact of this finding is that random-mapping BPCA, while providing the baseline performance, offers a more practical alternative to streaming PCA methods due to its parallel and scalable nature.

4.5.5 A Note on BPCA and Random-mapping

We showed in this section that the performance of mean-variance BPCA is significantly better compared to holistic PCA and the baseline random-mapping model. One question to investigate is to what extent can this be true for other applications? Based on our experiments in this chapter, we can say that BPCA can be significantly better in cases where the data attributes follow a multivariate normal distribution as the case in LFW and CelebA datasets. However, we will see later in Chapter 6 that for some type of data, BPCA provides almost the same reconstruction quality as the random-mapping model. Hence, BPCA provides a vary practical solution for a wide-range of applications where the data attributes follow a multivariate normal distribution. For other types of data, the efficacy of BPCA needs further investigation.

4.6 Conclusions

In this chapter, we proposed two partitioning-based PCA methods for modelling large-scale datasets. The main idea is to divide data attributes into smaller subsets and then apply PCA

Table 4.4: Reconstruction quality of BPCA when using the two mapping strategies and with different estimating subsets compared to the baseline random mapping.

	random mapping	**Mean-based mapping**			**Mean-variance-based mapping**		
Estimating subset size	-	**100**	**1000**	**full samples**	**100**	**1000**	**full samples**
compression ratio	12.5:1	12.6:1	12.6:1	12.8:1	12.7:1	12.6:1	12.6:1
MSE	0.0091	0.0028	0.0017	**0.0015**	0.0016	0.0009	**0.00074**
SSIM	0.43	0.72	0.79	**0.815**	0.79	0.86	**0.875**

to each partition separately. We showed that such models have many advantages over the standard holistic approach, including reduced reconstruction errors, increased parallelism and scalability. The first model, CPCA, was inspired by the JPEG compression standard where images are spatially divided into smaller blocks. We showed that reconstruction errors were significantly reduced when using only 20 cell eigenfaces in comparison to the reconstructions using the holistic eigenfaces. However, such a technique produces subtle cell boundary artifacts in the reconstructed samples, which can be reduced by increasing the cell size. The second model, BPCA, is more general in that it makes no spatial assumptions when mapping attributes but rather such mapping is based on the distribution of the values of attributes. We showed that mapping attributes using the mean and variance was better than mapping based only on the mean. Both mappings were shown to be superior to the random mapping model. We also found that the eigenfaces produced using random-mapping BPCA resemble the holistic eigenfaces. This suggests that the random-mapping BPCA gives the baseline performance which is analogous to the holistic PCA approach but entails lower memory costs and computation run time. Not only are our proposed methods beneficial for data compression, but they may also provide light-weight data representation for further machine learning tasks as will be explored in the next chapter. Investigating other partitioning strategies for non-normally distributed data is an interesting avenue for future research.

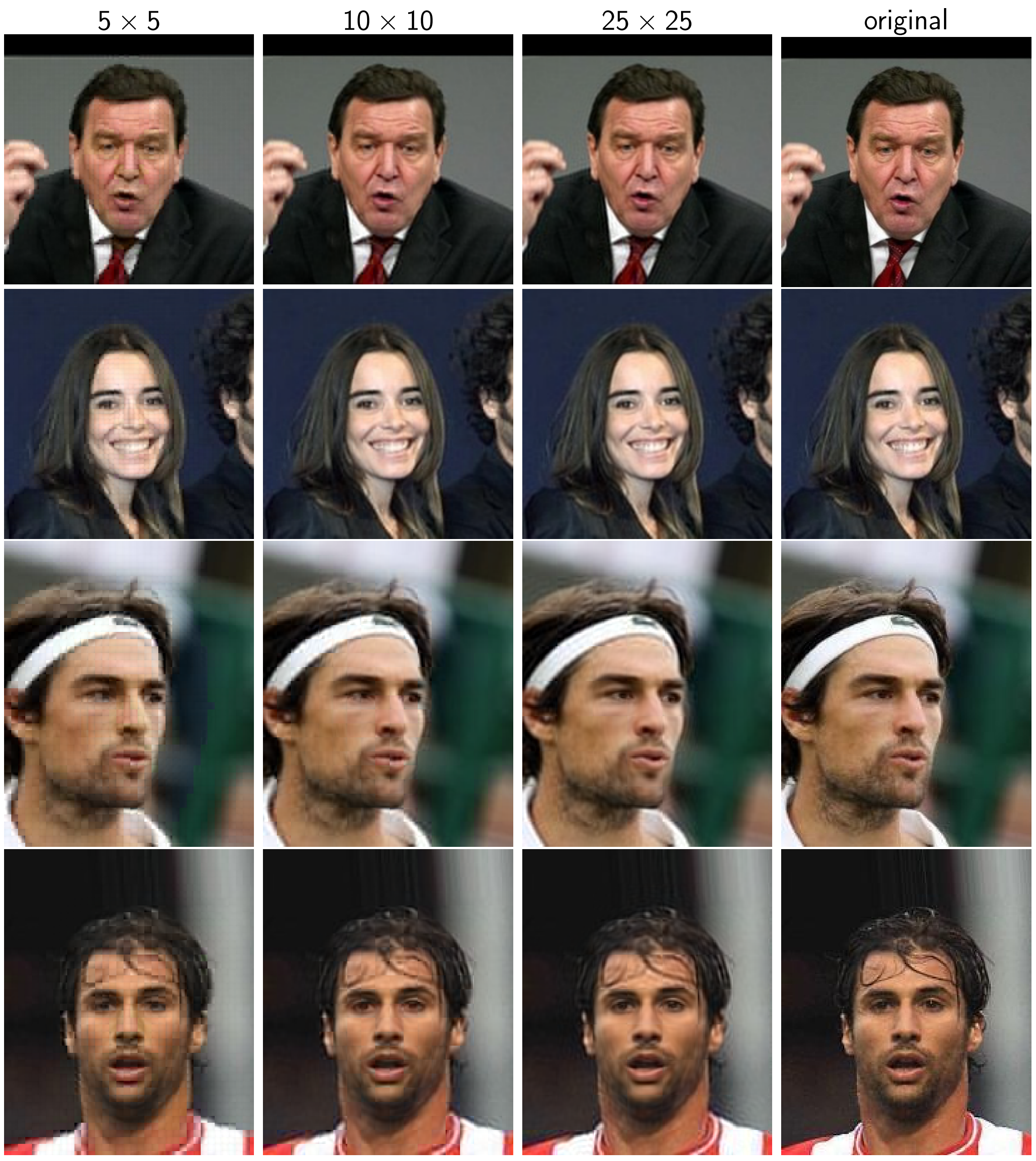


Figure 4.5: Reconstructed samples from LFW and CelebA when maintaining 15:1 compression ratio using cell eigenfaces of different cell-sizes. Note that block artefacts are more apparent when using smaller cell sizes.

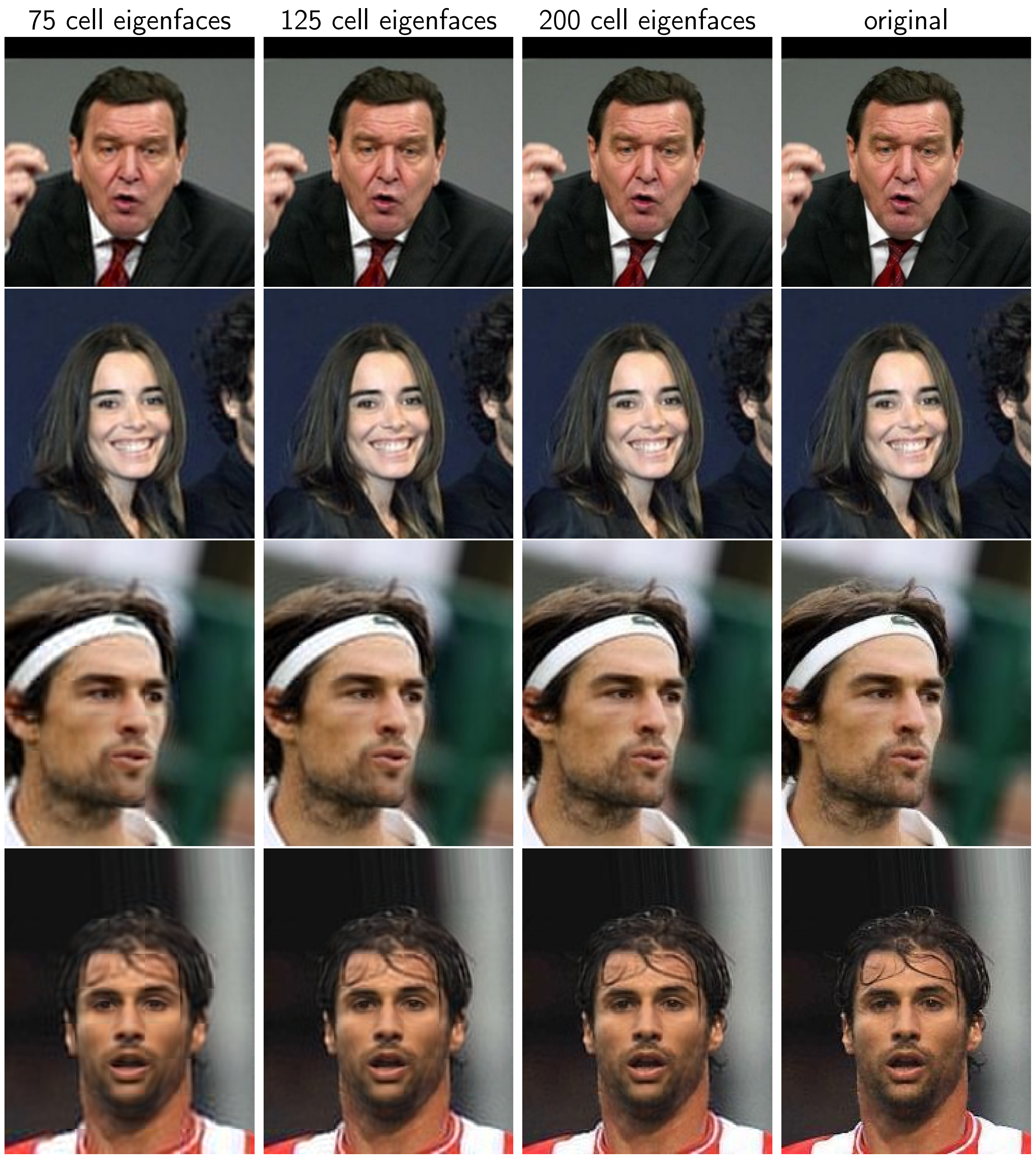


Figure 4.6: Reconstructed samples from LFW and CelebA when using different numbers of cell eigenfaces of cell-size 25 $\times$ 25. Note that the more number of cell eigenfaces used the better reconstruction quality achieved and the less apparent the block artefacts.

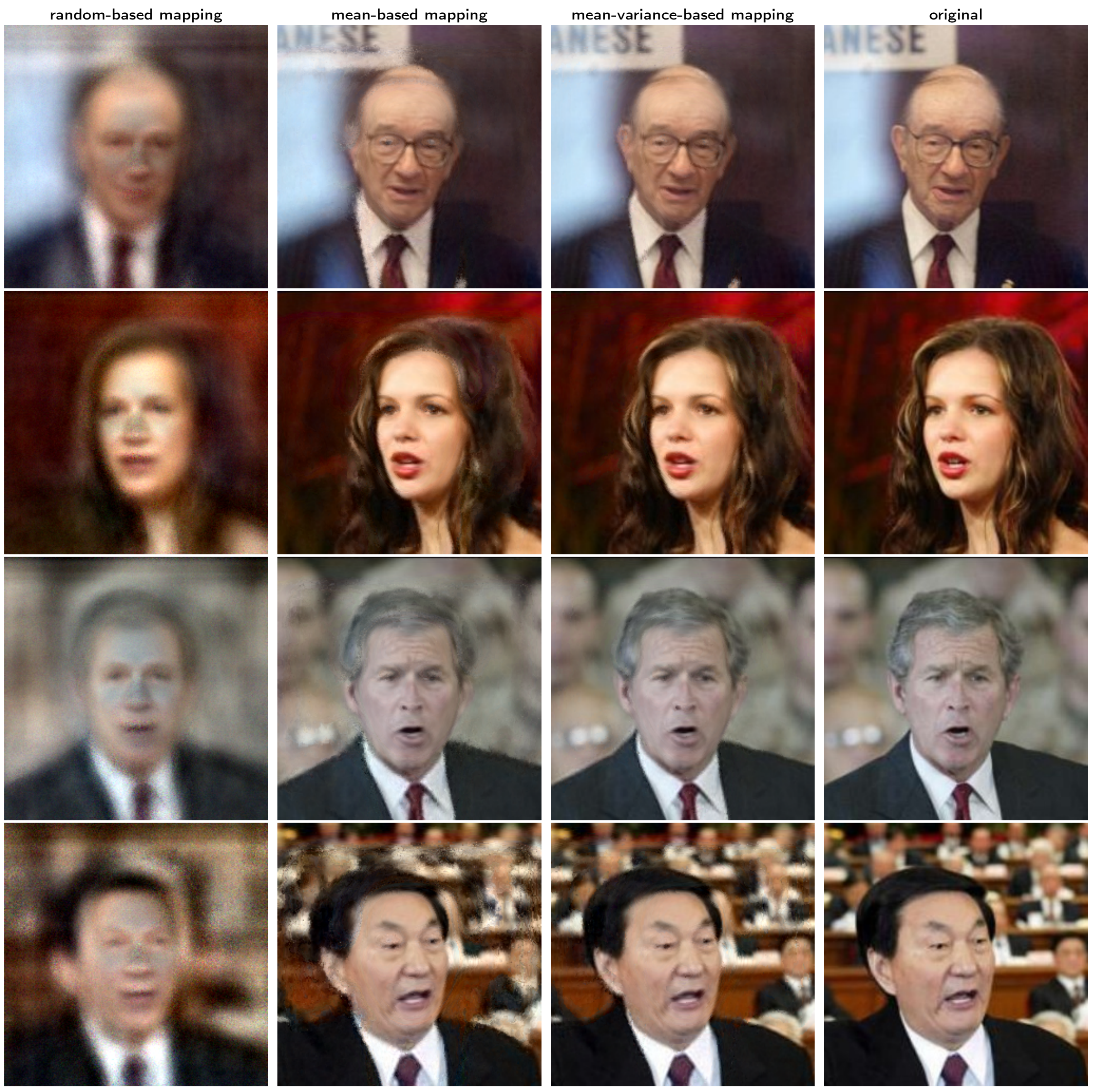
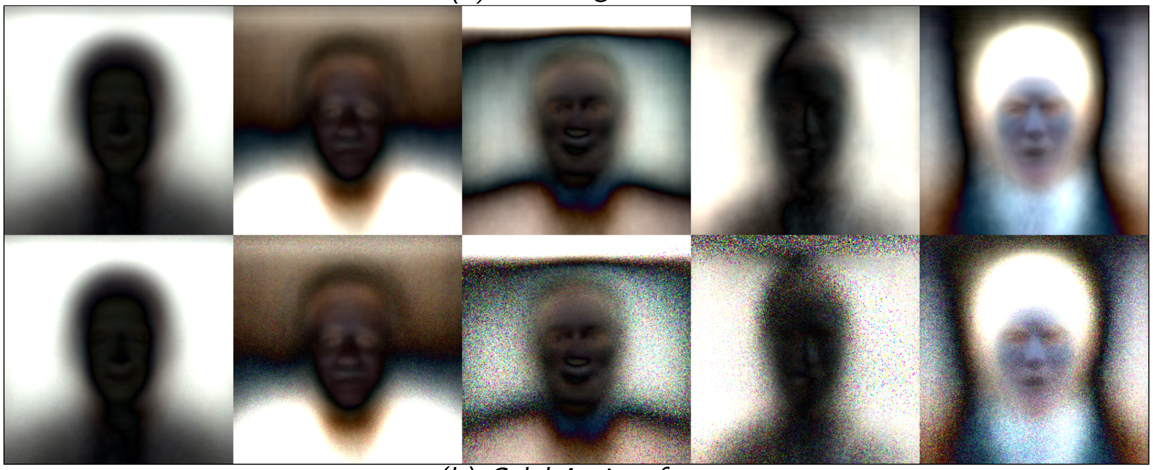


Figure 4.7: Reconstructed samples from LFW dataset using different mapping strategies for BPCA. Note that the mean-variance mapping provides a better reconstruction compared to mean-based mapping.

(a) LFW eigenfaces



(b) CelebA eigenfaces

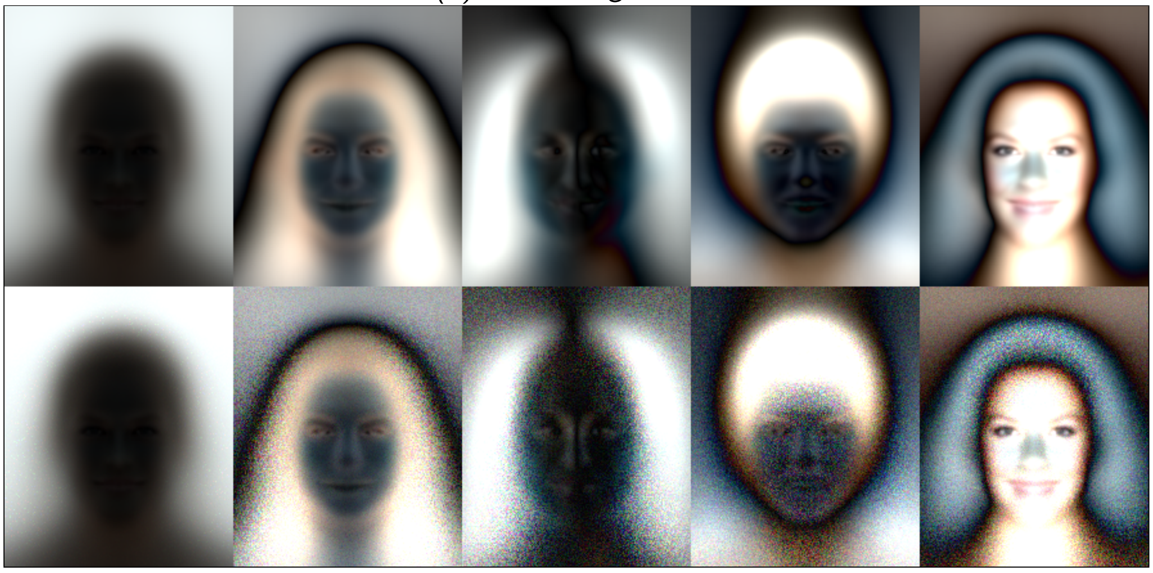
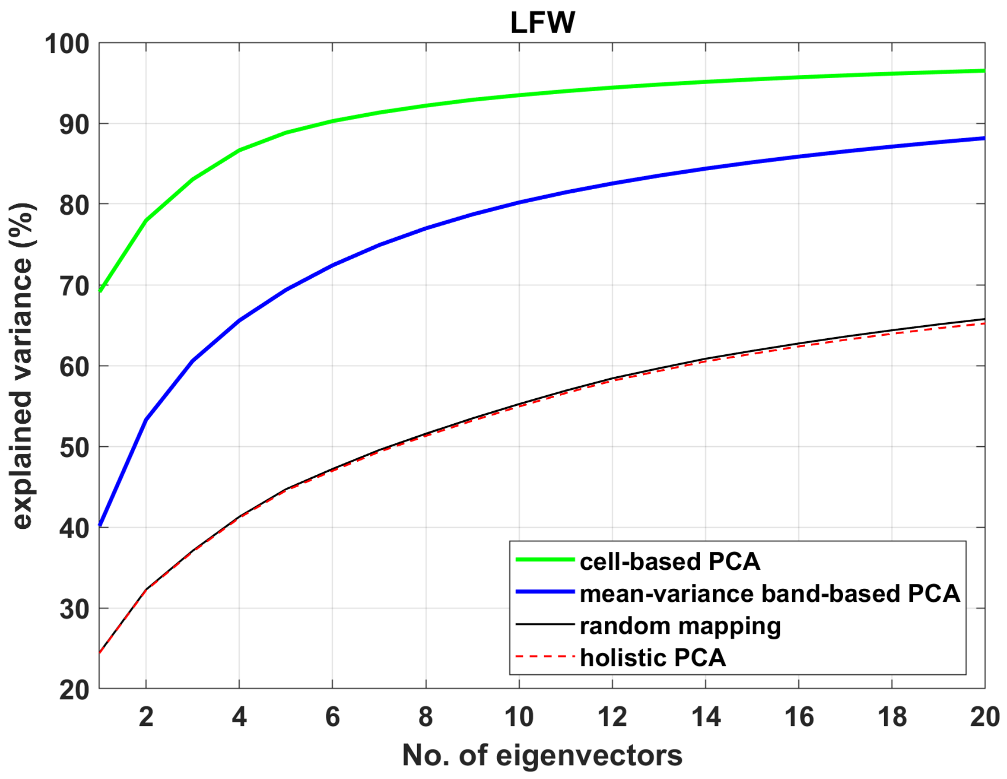


Figure 4.8: Comparing the holistic eigenvectors of LFW and CelebA computed using streaming PCA with the solution produced by baseline random-mapping model (bottom rows).



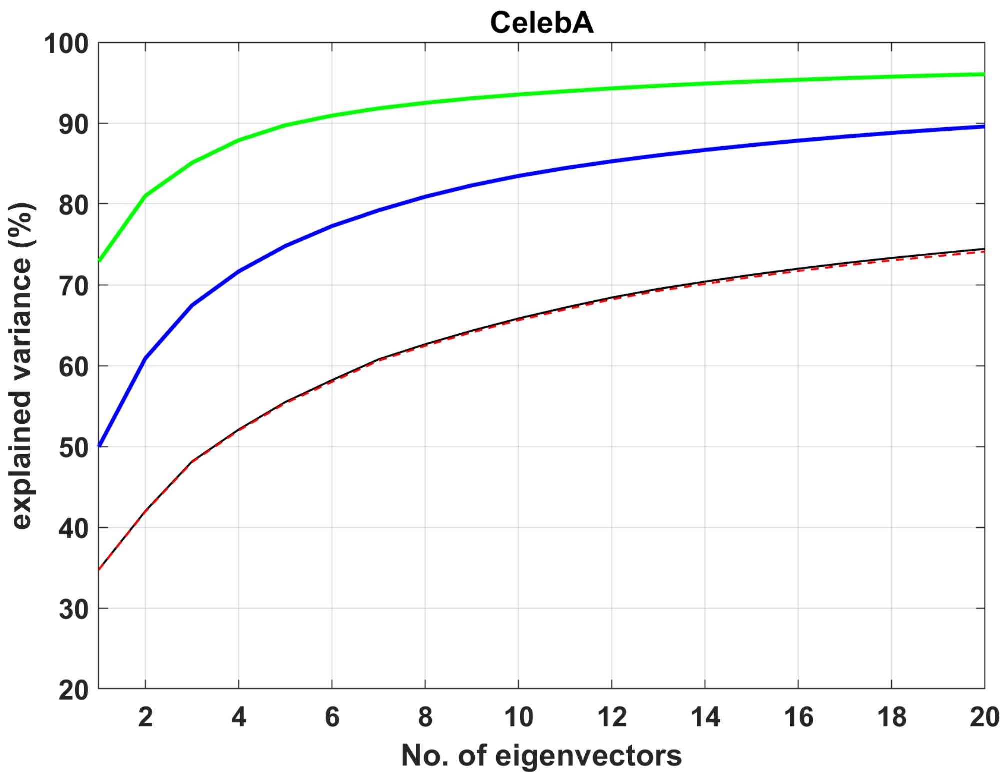


Figure 4.9: Comparison between the different mapping strategies with the holistic PCA in terms of the explained variance curve.

5 Learning from Limited and Reduced-representation Examples

In the previous chapter, we introduced two partitioning-based PCA models and highlighted their advantages over the traditional holistic scheme, including reduced run-time and reconstruction errors. In this chapter, we utilize CPCA for deriving lightweight representations that would lead to a better perception in supervised deep learning classifiers even when considering a limited number of training examples. By perception, we mean the classification accuracy of the trained model on the test samples. Our motivation for teaching such models using limited examples is to ensure more efficient learning using minimal information. We further develop a novel CNN classifier, RedNet, that can outperform state-of-the-art deep learning models when trained with limited examples. We compare the perception of RedNet when representing samples using CPCA and the conventional image downsampling representations.

5.1 Introduction

While PCA has been widely employed for classification tasks such as face recognition, the scale of datasets nowadays is beyond the computation constraints of typical modern machines to compute the eigenspace. In recent years, the use of neural networks has gained more attention in the machine learning community. In particular, deep convolutional neural networks are in rapid development in the field for many reasons. Firstly, they are much lower in computational cost compared to the fully-connected multi-layer perceptron (MLP) and can be easily parallelized using GPUs. In addition, they achieve state-of-the-art results in a wide range of learning tasks, including image classification and pattern recognition.

Despite the elegance of convolutional neural networks, the massive size of datasets nowadays remains one main hurdle that impacts the learning time of such models. For instance, the ImageNet dataset, one of the most influential datasets in the deep learning community, consists of 15 million high-resolution images of over 22,000 classes. A deep convolutional neural network may take up to months to be fully trained on such a dataset, not to mention the stochastic learning nature of these paradigms. In addition, state-of-the-art deep CNNs are trained with relatively large numbers of samples. For example, the CIFAR-10 dataset, one of the most widely used benchmarks for classifying low-resolution images, contains 50,000 training samples with 5,000 images per class. It would be thus interesting to ask: What would be the performance when training deep learning models using limited examples? This question is motivated by the fact that learning with minimal training examples implicitly entails a more efficient learning mechanism.

5.2 Objectives

The main objective of this chapter is to find efficient strategies for enhancing the perception of state-of-the-art deep learning classifiers considering a limited number of reduced-size examples. This requires investigating the two aspects of the problem. Firstly, one needs to reduce the size of the input samples in a way that preserves its key features. It is also essential to ensure that the reduced representation is spatially structured so that it is possible to apply deep CNN models. It is well-known that deep CNNs implicitly perform dimensionality reduction on the input samples. However, such an implicit process adds extra overhead computational cost and hence significantly increases the learning time. It is becoming interestingly common to downsample high-resolution image datasets in order to speedup the training of deep CNNs. Another possible solution is to employ deterministic dimensionality reduction schemes such as PCA or LDA. However, the fact that such schemes are considered to be holistic data representations makes applying such paradigms in conjunction with CNNs not appropriate. In addition, such a holism tends to focus mainly on global features while neglecting important local details. This concern also applies to fully-connected multi-layer perceptron models. It is very important also to consider that performing PCA on high-resolution images may be infeasible due to the quadratic computational dependence on data size. In order to address this problem, we utilize CPCA as a CNN-friendly dimensionality reduction scheme that efficiently reduces the size of the input samples while preserving their main features.

The second and more important aspect concerns the actual perceptual process in the CNN. In particular, how to boost the accuracy of the model given appropriately reduced but limited patterns? To the best of my knowledge, this important question has not been widely discussed in the literature. We investigate effective treatments for such a problem. We show that applying cross channel normalization in conjunction with grouped convolutional layers significantly enhances the accuracy. Based on these techniques, we develop a novel CNN model, RedNet, that is able to outperform state-of-the-art deep learning classifiers when considering limited training examples. Furthermore, we show that the accuracy of RedNet when representing samples using CPCA is better compared to the performance when downsampling images using conventional bicubic interpolation.

5.3 Related Work

There have been very important advances regarding the development of deep learning models for image classification. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) series is a yearly event that is specialized in evaluating such models using the ImageNet dataset [Rus+15]. In 2012, Alex Krizhevsky won the challenge and published one of the most influential papers in the area of convolutional neural networks [KSH12]. The model described in the paper was later referred to as AlexNet. This model has provided many contributions to the art of CNN design starting form the use of rectified linear units instead of the sigmoid functions, to the simulation of lateral inhibition in real neurons using cross channel normalization and ending with the use of grouping convolutional layers which effectively use the GPU resources to increase training efficacy. Despite being outperformed by other networks in accuracy, it remains one of the best CNNs in terms of learning speed. Simonyan and Zisserman proposed the VGG network in 2014 which mainly uses 3×3 convolutions in a way that increases the network depth while maintaining the computation [SZ14]. The GoogleNet, proposed by Szegedy et al., applies multi-scale processing and follows the Hebbian principle [Sze+15]. This increases both depth and breadth of the network. Both VGG and GoogleNet won the ILSVRC contest in 2014. One problem that arose with deep CNN is the vanishing gradient problem in which back-propagated gradients become close to zero due to the deep architecture. This prevents weights from being changed, especially in the first layers which may slow-down or even freeze the learning process. The idea of residual learning in neural networks received wide acclaim in the community for solving such a problem especially after ResNet won the 2015 challenge [He+16]. Residual networks are also analogous to the neurons in the human brain. Huang et al. generalized the concept of ResNet using Dense Convolutional Networks (denseNet) [Hua+17]. Zagoruyko and Komodakis studied the perception of residual architectures when increasing the number of feature maps and adding dropout layers in each residual block while decreasing the overall depth [ZK16]. More sophisticated architectures were later proposed based on the hybridization of the aforementioned models. For instance, ResNeXt, the winner of the 2016 challenge, combines ResNet and AlexNet architectures [Xie+17]. Other hybrid CNNs include Xception [Cho17], Inception-v3 [Sze+16] and Inception-ResNet-v2 [Sze+17]. It can be noted that deep CNNs in recent years tend to be much wider and deeper in architecture (with more than hundred convolutional layers) and hence more computationally expensive. In addition, state-of-the-art CNNs are tested with large number of samples. To the best of our knowledge, performance of such networks when using limited training examples has not been reported. From another perspective, while CNNs implicitly apply dimensionality reduction across layers through the use of pooling units, the role of deterministic dimensionality reduction approaches (such as PCA) in CNNs is still unclear. Could deterministic dimensionality reduction approaches lead to better perception in the CNN perspective? Chan et al. studied the use of convolutional filters that are computed using PCA and Linear Discriminant Analysis (LDA). The derived model was called PCANet. They found that such simple networks led to promising results when tested on many classification datatasets [Cha+15]. One main limitation of this technique, from a high level perspective, is the fact that the PCA filters used are computed based only on the input sample and not based on the whole dataset distribution. More recently, Gueguen et al. proposed using the coefficient (scores) in the JPEG-DCT space as light-weight representation for ResNet-50 model [Gue+18]. This led to improvement in accuracy with up to 1.3 speedup. Since the DCT space is derived per sample and does not depend on the distribution of the dataset, this model shares the same limitations as PCANet.

5.4 ConvNet vs MLP

In Section 2.4.1 in the background chapter, we introduced the main mechanism behind the fully-connected feed-forward MLP learning using the back-propagation algorithm. In this section, we discuss the basics of convolutional neural networks and their main differences from MLPs. Unlike MLP, where each unit in a layer depends on the entire units in the previous layer, CNN regularizes spatial patterns using a moving window type of processing. In this case, each unit in a layer depends on units in the previous layer within a specific patch. The moving window consists of multiple filters (also referred to as kernels or convolutions) where each filter contains weights for the corresponding patch. The window moves through the entire image with a specific stride. After computing the convolutions, the activation function is performed followed or preceded by a normalization process. The activation function is usually a rectified linear unit (ReLU) defined as follows $f(x) = \max(0, x)$. Such activation was found to be advantageous compared to the sigmoid function in terms of minimizing the loss function [KSH12]. It has become a common practice to use normalization layers in CNNs. There are two main types of these layers that will be discussed more in section 5.5.2. Pooling layers may also be used in CNNs. These layers operate a moving window in which a single value is chosen per patch per kernel. Two types of pooling layers are commonly used, max pooling and average pooling. The output of each convolutional layer is referred to as a feature map. The final layers usually follow the structure of fully-connected MLP. Like the conventional MLP, CNNs are also trained using the back-propagation learning algorithm in which convolutional gradients are derived using the chain rule. Figure 5.3 shows a simple CNN structure consisting of one convolutional layer, one pooling layer and fully-connected MLP layers.

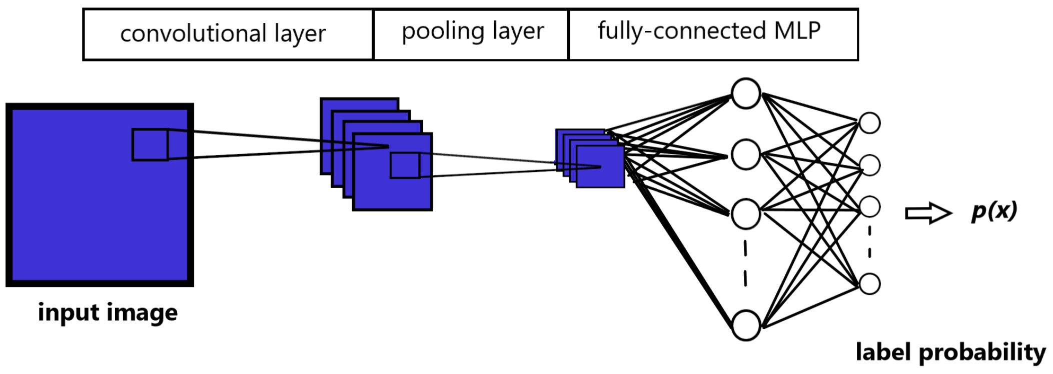


Figure 5.1: A simple CNN structure.

5.5 RedNet: A Deep Learning CNN for Reduced-size Patterns

In this section, we describe the structure of our deep CNN. The network works on low-resolution images and is inspired by the AlexNet model. The main difference between our network and AlexNet is that we use cross-channel normalization after each convolutional layer, whereas in AlexNet there only exist two cross-channel normalization layers. We also use different parameters for such layers.

5.5.1 Cross Channel Normalization

For the normalization layers, there are two main types that are usually used: Batch normalization and cross channel normalization (also known as local response normalization). The batch normalization is the most commonly used in the literature [IS15]. It basically normalizes each value in the feature map as follows

$$x_{i,(x,y)}^{norm} = \frac{x_{i,(x,y)} - \bar{x}_{i,(x,y)}^{B}}{\sqrt{\left(\sigma_{i,(x,y)}^{B}\right)^2 + \epsilon}},$$

where $\bar{x}^B$ and σ^B are the mean and variance of mini-batch B respectively and $i,(x,y)$ defines the value location within the feature map. The cross channel normalization layer, while not widely used, has a strong resemblance to the human biological neural system. It simulates the lateral inhibition phenomenon where excited neurons subdue their neighbors in order to allow for increased sensory perception. Cross channel normalization is usually placed after ReLU activation units. In our model, we applied cross channel normalization using the following formula

$$x_{i,(x,y)}^{norm} = \frac{x_{i,(x,y)}}{\left(K + \frac{\alpha}{n_w} \sum_{j=\max(1,i-n_w/2+1)}^{\min(N_{channel},i+n_w/2)} x_{j,(x,y)}^2\right)^{\beta}}$$

where α, K and β are hyperparameters and n_w is the window size and $N_{channel}$ is the number of kernels in the current layer. We set the window size to 16, $K = 1$, $\alpha = 0.02$ and $\beta = 0.75$. We found that such biologically inspired normalization significantly boosts accuracy in comparison to the batch normalization as will be discussed in the results section. However, the main downside of such a strategy is the expensive computational cost, specifically when using large window sizes.

5.5.2 Grouped Convolutional Layers

Grouped convolutional layers were first introduced in Krizhevsky's paper [KSH12]. They efficiently use GPU resources to create multiple learning paths in order to enhance classification accuracy. The efficiency of such a technique was further discussed by Xie et al. in their ResNeXt model, where each residual block contained 32 learning paths [Xie+17]. This has pushed state-of-the-art results on a number of benchmarks including CIFAR-10 and ILSVRC 2016 challenge. In our model, we employed two learning paths using grouped convolutional layers. We found that such a technique enhances the average accuracy according to our experiments.

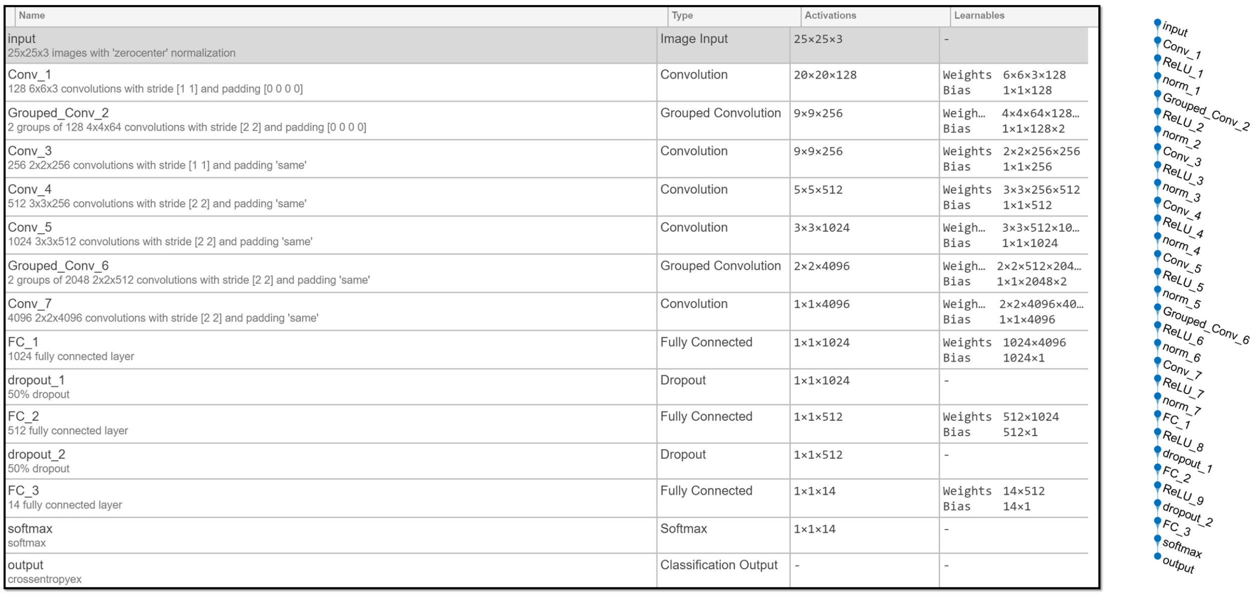
5.5.3 Overall Structure

Figure 5.2 shows the detailed structure of our CNN. We refer to this network as Reduced Net (RedNet). One can note that there are no pooling layers in the network. We reduce the feature map size across layers by applying convolutions with strides greater than one and setting the padding to zero in most layers. Dropout layers are used in order to regularize network performance. The initial network weights were set according to Glorot initializer [GB10]. The input size of the network is set to 25×25. We test the network when reducing the size of the original images using bicubic interpolation. We then compare the performance when using the CPCA representation. The image pixel values were represented as doubles in the range of $[0, 255]$. We found that such range of values give better classification accuracy. Other state-of-the-art models use [0, 1] pixel representations. Hence, in the results section, we use the first representation for RedNet and the second $[0, 1]$ representation for the other state-of-the-art models.

5.6 Experiments

5.6.1 Main Task

As a testbed, we consider the task of recognizing faces from LFW dataset. One problem in this dataset is that the number of samples per subject is limited and not uniform. In fact, most subjects have fewer than 10 samples. We conduct our empirical evaluation on the subjects



Name	Type	Activations	Learnables
input 25x25x3 images with 'zerocenter' normalization	Image Input	25×25×3	-
Conv_1 128 6x6x3 convolutions with stride [1 1] and padding [0 0 0 0]	Convolution	20×20×128	Weights 6×6×3×128 Bias 1×1×128
Grouped_Conv_2 2 groups of 128 4x4x64 convolutions with stride [2 2] and padding [0 0 0 0]	Grouped Convolution	9×9×256	Weigh… 4×4×64×128… Bias 1×1×128×2
Conv_3 256 2x2x256 convolutions with stride [1 1] and padding 'same'	Convolution	9×9×256	Weights 2×2×256×256 Bias 1×1×256
Conv_4 512 3x3x256 convolutions with stride [2 2] and padding 'same'	Convolution	5×5×512	Weights 3×3×256×512 Bias 1×1×512
Conv_5 1024 3x3x512 convolutions with stride [2 2] and padding 'same'	Convolution	3×3×1024	Weigh… 3×3×512×10… Bias 1×1×1024
Grouped_Conv_6 2 groups of 2048 2x2x512 convolutions with stride [2 2] and padding 'same'	Grouped Convolution	2×2×4096	Weigh… 2×2×512×204… Bias 1×1×2048×2
Conv_7 4096 2x2x4096 convolutions with stride [2 2] and padding 'same'	Convolution	1×1×4096	Weigh… 2×2×4096×40… Bias 1×1×4096
FC_1 1024 fully connected layer	Fully Connected	1×1×1024	Weights 1024×4096 Bias 1024×1
dropout_1 50% dropout	Dropout	1×1×1024	-
FC_2 512 fully connected layer	Fully Connected	1×1×512	Weights 512×1024 Bias 512×1
dropout_2 50% dropout	Dropout	1×1×512	-
FC_3 14 fully connected layer	Fully Connected	1×1×14	Weights 14×512 Bias 14×1
softmax softmax	Softmax	1×1×14	-
output crossentropyex	Classification Output	-	-

Figure 5.2: Details of the RedNet layers.

associated with largest number of samples. For each subject, we choose a fixed number of training and testing samples. We evaluate performance based on a number of trials where in each trial the training set and test set are selected randomly. Two experimental settings are considered. The first one, LFW-14-subject, considers 14 subjects of which each has 45 training images and 3 test images. The second setting, LFW-24-subject, takes 24 subjects with 30 training images and 5 testing images per subject. The LFW dataset addresses the problem of identifying the face in uncontrolled environment. While there are many enhanced versions of the LFW dataset where images are deep-funnelled in order to limit face variations and improve the recognition rate, we used the original version in our experiments. We choose the LFW dataset for evaluation because of the fact that the unconstrained face recognition problem is one of the best examples for such type of problems where subject samples are limited and low in resolution; for instance, faces in a CCTV stream. Other datasets such as CIFAR-10 and ImageNet are more suited for cases where the training samples are immense; for instance, we can obtain in practice thousands of images of frogs, buses, etc. That said, the unconstrained face recognition problem should not be taken as an easier task, and that from a practical perspective, this can be of significance to many real-world applications.

5.6.2 Results

We first evaluate the performance of RedNet when representing images using CPCA and bicubic interpolation. We use LFW-14-subject in this evaluation. Figure 5.3 shows 10 samples for each subject downsampled to 25 $\times$ 25 pixels using bicubic interpolation [Key81] where the amount of variations in face appearance can be clearly noted even for the same subject . We run 100 trials and report the statistics on the test set. As mentioned earlier, for each trial, the

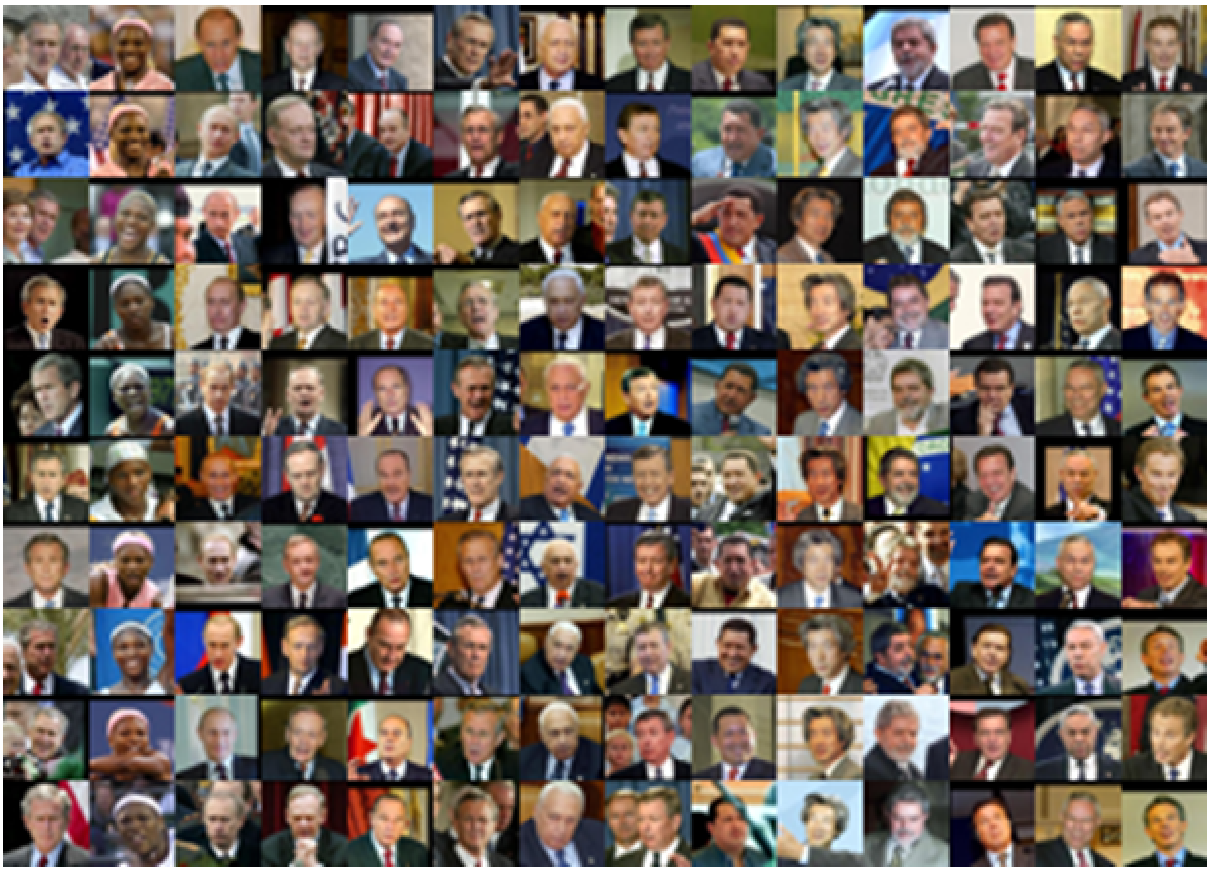


Figure 5.3: Downsampled face images using bicubic interpolation from 250×250 pixels down to 25×25 pixels. This depicts the main challenges that can occur in an image including scale, rotation, expression, illumination and background variations.

training set and test set are chosen on a random basis. The bicubic interpolation is performed on both grayscaled and colored images producing images of size $25 \times 25 \times 1$ and $25 \times 25 \times 3$ respectively. For CPCA, the cell-size is set to 10×10. This results in a reduced representation of size $25 \times 25 \times n_{eig}$ where n_{eig} is the number of eigenvectors per block. We studied the CPCA representation for different numbers of n_{eig}, namely, $n_{eig} \in \{1, 2, 3, 4, 10\}$. The cell eigenfaces are computed based on the full LFW dataset. For each representation, we train the RedNet model using 200 epochs with a mini-batch size of 128. The learning algorithm is stochastic-gradient-descent with momentum of 0.9 and initial learning rate of 0.008. The learning rate is halved every 100 epochs. In all experiments in this chapter, we applied ℓ^2-norm regularization with a factor of 0.0001. We do not apply any data augmentation in the training phase. However, we applied data shuffling after each epoch. We also study the perception of RedNet when replacing the Cross channel normalization units with batch normalization layers in order to show how such a technique can significantly enhance the accuracy. In the case of batch normalized RedNet, we place the normalization layers before the ReLu units as done in the literature.

Table 5.1 reports the statistics. It is evident that when representing images using CPCA, the

Table 5.1: RedNet classification results on LFW-14-subject when employing CPCA in comparison to bicubic downsampling.

Model	Batch Norm RedNet (CPCA)	RedNet (CPCA)					RedNet (bicubic)	
Input Size	$25 \times 25 \times 3$	$25 \times 25 \times 1$	$25 \times 25 \times 2$	$25 \times 25 \times 3$	$25 \times 25 \times 4$	$25 \times 25 \times 10$	$25 \times 25 \times 1$ (grayscale)	$25 \times 25 \times 3$
Mean Accuracy	58.6%	73.1%	75.95%	**76.45%**	75.36%	75.43%	67.48%	73%
Standard Deviation	7.34%	6.23%	6.14%	**5.7%**	6.77%	5.93%	7.23%	5.78%
Mode	54.76%	71.43%	**76.19%**	73.81%	73.81%	**76.19%**	71.43%	71.43%
Mode Probability	0.15	0.18	**0.19**	0.17	0.15	0.18	0.15	0.2
Pr(*accuracy* $> 70\%$)	0.04	0.69	0.83	**0.87**	0.78	0.79	0.37	0.72
Pr(*accuracy* $> 80\%$)	0	0.13	0.26	**0.31**	0.28	0.25	0.05	0.12
Max Accuracy	78.57%	88.1%	88.1%	88.1%	**90.48%**	88.1%	85.71%	85.71%
Min Accuracy	42.86%	59.52%	**61.9%**	59.52%	59.52%	**61.9%**	47.62%	57.14%
No. of Epochs/trial	400	200						
Mini-batch Size	128							
No. of trials	100							

accuracy of RedNet (when applying cross channel normalization) is significantly enhanced in all statistical aspects. The network gives the best performance in terms of the mean accuracy when using three eigenvectors per block $n_{eig} = 3$. One can also note that in case of only a single cell eigenface $n_{eig} = 1$, the network accuracy is better compared to the grayscale downsampling. However, other than the slightly enhanced marginal performance, there is no notable improvement when increasing the number of eigenvectors per block over 3 cell eigenfaces. One can also note that the batch-normalized version of RedNet gives very poor results despite using the CPCA representation and being trained with more epochs. This suggests that the cross channel normalization gives remarkably better perception especially when dealing with limited training examples.

We now compare the results of RedNet with state-of-the-art models trained from scratch using high-resolution images from LFW-14-subject. In particular, we examine the perception of AlexNet, GoogleNet, 18-layer ResNet, Inception-v3 and Xception. For each model, images are resized using bicubic interpolation to fit the size of the corresponding input layer. We follow the training settings specified in the original papers as much as we can. However, for GoogleNet we adapt the settings to work properly with the dataset so that the loss function is minimized better in the training phase. The mini-batch size is chosen based on the computation capability. Experiments are conducted on a machine with a GeForce GTX 970 GPU. For state-of-the-art models, weights are initialized as specified in the original papers. Table 5.2 shows the results on the test set as well as the training settings. It can be noted that RedNet achieves the best results in comparison to state-of-the-art models even when representing images using the bicubic downsampling. On the other hand, state-of-the-art models give much lower accuracy despite being trained with higher-resolution images. Excepting AlexNet, all models take remarkably longer training time when considering the same number of epochs (200 epochs).

Lastly, we evaluate the accuracy with deep models utilized for low-resolution images of size $32 \times 32 \times 3$. Specifically, we compare performance with 56-layer ResNet, Wide-ResNet of depth 28 and width 10, and last but not least DenseNet with network growth of 24 and depth

Table 5.2: Comparing results of state-of-the-art models trained from scratch using high-resolution images from LFW-14-subject with the results of RedNet.

Model	RedNet (CPCA)	RedNet (bicubic)	AlexNet	GoogleNet	ResNet-18	Inception-v3	Xception
Input Size	$25 \times 25 \times 3$	$25 \times 25 \times 3$	$227 \times 227 \times 3$	$224 \times 224 \times 3$	$224 \times 224 \times 3$	$229 \times 229 \times 3$	$224 \times 224 \times 3$
No. of trials	100	100	100	100	30	10	10
Mean Accuracy	**76.45%**	73%	58.98%	64.83%	44.37%	65.48%	54.5%
Standard Deviation	**5.7%**	5.78%	7.3%	10.18%	11%	9.41%	14.95%
Mode	**73.81%**	71.43%	54.76%	64.29%	50%	73.81%	50%
Mode Probability	17/100	20/100	14/20	14/100	5/30	4/10	3/10
Max Accuracy	**88.1%**	85.71%	73.81%	85.71%	69.05%	73.81%	**88.1%**
Min Accuracy	**59.52%**	57.14%	40.48%	30.95%	21.34%	47.68%	42.86%
Mini-batch Size	128	128	128	64	128	32	32
Training Time	13.2 minutes	13.2 minutes	12 minutes	27.35 minutes	25 minutes	8.5 hours	9.9 hours
No. of Epochs	200	200	300	200	200	200	200
Training Settings	**Learning:** SGDM **Momentum:** 0.9 **Initial lr:** 0.008 **Drop factor:** 0.5 **Drop period:** every 100 epochs	**Learning:** SGDM **Momentum:** 0.9 **Initial lr:** 0.008 **Drop factor:** 0.5 **Drop period:** every 100 epochs	**Learning:** SGDM **Momentum:** 0.9 **Initial lr:** 0.0008 **Drop factor:** 0.5 **Drop period:** every 100 epochs	**Learning:** SGDM **Momentum:** 0.9 **Initial lr:** 0.002 **Scheduler:** constant learning rate	**Learning:** SGDM **Momentum:** 0.9 **Initial lr:** 0.1 **Drop factor:** 0.1 **Drop period:** every 60 epochs	**Learning:** RMSprop **square gradient decay rate:** 0.9 **Initial learning rate:** 0.045 **Drop factor:** 0.94 **Drop period:** every 2 epochs **Gradient threshold:** 2 using ℓ^2-norm	**Learning:** SGDM **Momentum:** 0.9 **Initial lr:** 0.045 **Drop factor:** 0.94 **Drop period:** every 2 epochs

of 100. In order to have a better idea of the performance when increasing the number of subjects and reducing the number of training samples, this evaluation is based on the LFW-24-subject where each subject has 30 training images and 5 test images. Table 5.3 report the results. It is apparent that the RedNet accuracy is much better compared to the other models. In addition, it can be noted that despite the decline in RedNet average accuracy from 76.54% to 67.52% when increasing the number of subjects and decreasing the number of training samples, the minimal accuracy achieved is interestingly very close to the one resulted from LFW-14-subject.

5.7 Conclusions

In this chapter, we investigate how to efficiently train deep learning models for classification tasks using limited and reduced-size training examples. We propose a novel deep CNN model, RedNet, that is inspired by AlexNet but uses more cross channel normalization layers. We show that such a biologically inspired process significantly enhances the perception of the model in such scenarios. We also show the efficacy of CPCA as a CNN-friendly dimensionality reduction scheme that is able to reduce the size of the training samples while preserving their main features. Experiments on LFW-14-subject and LFW-24-subject show that the classification results of RedNet are always better when using CPCA instead of the traditional downsampling technique. In addition, we show that state-of-the-art deep convolutional neural networks, including ResNet, DenseNet, WResNet and Inception-v3 achieve poor results in such scenarios compared to RedNet. In terms of future work, it would be interesting to investigate non-linear representation paradigms such as cell-based kernel PCA.

Table 5.3: Comparing results of state-of-the-art models trained from scratch using 32×32 resolution images from LFW-24-subject with the results of RedNet.

Model	**RedNet (CPCA)**	**RedNet (bicubic)**	**ResNet-56**	**Wide-ResNet** 28×10	**DenseNet-100** ($k = 24$)
Input Size	$25 \times 25 \times 3$	$25 \times 25 \times 3$	$32 \times 32 \times 3$	$32 \times 32 \times 3$	$32 \times 32 \times 3$
No. of trials	100	100	100	40	30
Mean Accuracy	**67.52%**	65.83%	42.03%	54.5%	49.28%
Standard Deviation	**3.96%**	3.9%	4.78%	4.3%	5.57%
Mode	**66.67%**	65.83%	40.83%	52.5%	52.5%
Mode Probability	11/100	19/100	11/20	6/100	3/30
Max Accuracy	**76.67%**	74.17%	57.5%	63.3%	59.17%
Min Accuracy	**58.33%**	55.83%	29.17%	45%	38.33%
Mini-batch Size	128	128	128	128	64
Training Time	16.2 minutes	16.2 minutes	8.5 minutes	1.4 hours	3.3 hours
No. of Epochs	200	200	300	200	200
Training Settings	**Learning:** SGDM **Momentum:** 0.9 **Initial lr:** 0.008 **Drop factor:** 0.5 **Drop period:** every 100 epochs	**Learning:** SGDM **Momentum:** 0.9 **Initial lr:** 0.008 **Drop factor:** 0.5 **Drop period:** every 100 epochs	**Learning:** SGDM **Momentum:** 0.9 **Initial lr:** 0.1 **Drop factor:** 0.1 **Drop period:** every 60 epochs	**Learning:** SGDM **Momentum:** 0.9 **Initial lr:** 0.1 **Drop factor:** 0.2 **Drop period:** every 60 epochs	**Learning:** SGDM **Momentum:** 0.9 **Initial lr:** 0.1 **Drop factor:** 0.1 **Drop period:** every 150 epochs

6 Modelling Time-varying Systems

Up to this point, we considered large-scale datasets where samples are mutually independent. In this chapter, we address the problem of modelling time-varying systems where each sample depends on the previous time-steps. Analyzing time-varying data has been of significant importance to many scientific fields. In particular, this chapter delves in detail in the area of modelling physical time-varying observations due to the importance of designing efficient low-complexity PCA methods for such a type of applications and because the work of this PhD was primarily motivated by this area of study. What makes time-varying data analysis more challenging is the lack of benchmark evaluation strategies, particularly, the existence of benchmark test datasets. Furthermore, time-varying data can appear in many different types of dynamics and formats for which processing may vary significantly. The contribution of this chapter is two-fold. We first show the importance of using PCA for analyzing a wide array of time-varying systems by decomposing the system behaviour into a number of key dynamic modes. In addition, since standard PCA is not well scalable in terms of data size and dimensionality, we investigate the performance of streaming and partitioning-based PCA for such types of problems.

6.1 Introduction

Time-varying systems appear naturally in our world from the hourly weather readings to the population growth and global warming. Understanding the behaviour of such systems has been of paramount significance to the development of many scientific fields. Many physical phenomena, if not most, are described using time-dependent mathematical models. For instance, Newton's laws of motion, Schrödinger and Dirac equations which are amongst the most well-known landmarks in Physics are all indeed time-dependent models. In general, finding the time-dependent model that best describes a set of observations is not an easy task. One can find such models analytically having a deep understanding of the studied problem. However, the analytic solution of the input observations may not be possible due to the large size of the observation space (parameter space). As the typical parameter space is growing in size, designing efficient automatic modelling schemes for understanding the behaviour of the underlying systems is in high demand. Such a problem is challenging from many perspectives.

Firstly, such scientific simulations have different types of dynamics (stationary, harmonic, chaotic, etc) and data representations. The analysis methodology of each type differs from the other. In addition, due to the fact that modern datasets are very large in size, where time-varying data is no exception, the desire is more on designing scalable models.

In this chapter, we consider time-varying systems that are based on the wave equation. Such a model can describe the dynamics of a wide range of physical phenomena. We show that the solution derived by PCA to such systems is analogous to the analytic solution. This is of importance when finding the analytic solution is not feasible due to the streaming nature and large-scale of the data on hand. Furthermore, due to the fact that PCA has quadratic complexity dependence on data size, it may even not be possible to find the solution using the standard PCA, a problem that has not been well-explored in the literature. In order to solve this problem, we employ the techniques proposed in the previous chapters, namely streaming PCA and partitioning-based PCA and investigate their performance in terms of complexity, scalability and quality.

6.2 Related Work

Time-varying data analysis has gained more attention in the scientific visualization community. Since most scientific simulation data are massive in scale, most research emphasis is on designing efficient ways for extracting meaningful patterns from such simulations in order to better understand their dynamics. One common approach for analyzing large-scale time-varying volumetric data is to subdivide the volume into subblocks based on some strategies. Wang et al. proposed an importance-driven approach for time-varying volumetric data visualization [WYM08]. The approach subdivides the volume into blocks and then prioritize these blocks based on their importance curves. These importance curves were further used to detect abnormal events in the data. Chen et al. proposed a method for efficiently extracting spatio-temporal correlation patterns in order to analyze large-scale multivariate time-varying data [Che+11]. Borrowing techniques from the area of video compression and encoding, Jang et al. used basis function representations for compressing time-varying volumetric data [JEG12]. Other techniques for visually understanding time-varying data are based on the ways cartoonists depict motion in a single illustration such as speed-lines. Joshi and Bheingaus investigated the use of such techniques to better visualize time-varying data [JR05].

The use of PCA for modelling time-varying systems has been well-investigated in the biological physics community. In particular, PCA has been widely applied for identifying domain collective motion in biomolecular systems as an alternative to Normal Mode Analysis (NMA). Such a collective motion provides key insight into the biological functionality of these systems. Examples of such biomolecular systems include proteins [Vin09], enzymes [KW10] and nucleic acids [Oro+03], which all form the building block of our life. This collective type of dynamics is not limited to biomolecular systems. In fact, non-biological molecules vibrate in a very similar manner that can also be described using NMA. The study of such vibrations is of potential importance to many fields in Physics including Carbon nanotubes technology [Rao+97; SXZ12] and neutrino oscillation [Air+18].

Studying the vibrations of polyatomic molecules (molecules with more than two atoms) by means of normal modes is a rather old technique dating back to 1940s when a comprehensive review of the topic was provided by the Nobel Prize laureate Gerhard Herzberg [Her45]. This involves solving an eigenvalue problem of which the number of solutions has quadratic dependence on the number of atoms in the studied molecule. The main problem is that finding the analytic solution in this way may be not feasible if the number of atoms is too large. It was not until the pioneer work of Trion [Tir96] when a simplified reduced parameter model was proposed in place of the memory and computational demanding traditional paradigm. The idea was further developed to what is known as the Elastic Network Model (ENM) and Gaussian Network Models (GNM) [BAE97; Ati+01; YSJ09]. In general, these techniques aim to reduce the size and complexity of the system using the concept of coarse-grained modelling which treats the whole molecular system by means of simpler subsystems instead of considering all individual atoms in the molecule [Ing+14; Kmi+16]. In many cases, this can be achieved by considering specific types of atoms (usually the alpha Carbon atoms C_α) and neglecting the remaining ones [DS10].

The main limitation of Normal Mode Analysis and Elastic Network Models is the fact that they are only capable of finding harmonic oscillations in the system. It has been observed that protein dynamics reveal a large degree of anharmonicity [RA09] which cannot be identified using NMA. The first reported use of PCA for modelling both types of behaviour in protein dynamics is credited to the work of Amasei et al. [ALB93], although the technique was termed *Essential Dynamics*, in which two subspaces were used to describe atomic fluctuations from their equilibria: The bases derived by PCA defining the essential degree of freedom that can describe most variations in molecule dynamics and the distribution of scores of time-steps in this reduced-dimensional space. Since then, PCA has been extensively used to find key collective motions of biomolecules as an alternative to NMA and ENM [HKG94; MR05; Yan+09; Bah+09; Skj+14]. Many molecular dynamics open-source libraries and programs offer built-in functions and tools for applying PCA to MD trajectories and visualizing the resulting dominant collective motions [Mon04; Cas+05; BMB11; McG+15]. In addition to finding the dominant behaviour of the biomolecule, PCA has also been applied for estimating the entropy of the actual biomolecule system from the MD simulation time-steps [BHM09; BGH06; AK01; TB04; SMG00] which is very important for understanding a wide variety of aspects of the studied biological system. Such analyses have led to a better understanding of the development of many complex diseases including HIV [Bri+05; BB09; IM09; BMS14] and different epigenetic diseases [BV12] and their reactions to different types of drugs. From a higher-level perspective, PCA is not limited to specific types of data format and does not make any spatial assumption of the data on hand. Cohen and Moerner [CM07] showed that when applying PCA to fluctuations of a single molecule DNA represented as a series of video frames, the solution was still very similar to the analytic solution.

One main question regarding the use of PCA for analyzing time-varying data is related to the appropriate time-scale and sampling frequency. Specifically, what is the appropriate time-scale and sampling frequency in order to converge to the actual significant modes of the analytic model? Since the most significant eigenvectors of given observations correspond to lower frequency eigenmodes, they reflect the long-term behaviour of the system. Therefore, it is necessary to ensure that the observations included cover a long enough time-scale. However, in practice, it is very common that the lowest eigenfrequency is unknown. Furthermore, such eigenmodes covers wide frequency band in which the difference between the highest and lowest frequency is relatively large. This necessitates that the sampling rate and time-scale of observations should both be large in order to guarantee convergence to the analytic solution. For instance, key biological functions in proteins range from femtosecond time-scale to the the order of seconds [Vos+91; Cla+95; Bal+96; MLS09; RA09], not to mention the large number of atoms (high dimensionality) in such molecules. Thus, the diagonalization of the resulting covariance matrices (or dual covariance matrices) is usually beyond the computational capability of typical modern machines. Hence, it is important to study the convergence rate of streaming PCA for such dynamical systems. Typically, for a set of time-varying observations, a streaming PCA should provide a reliable estimation of the actual eigenvectors up to the recent time-step. To the best of our knowledge, the performance of streaming PCA when analyzing such practical use-cases has not been investigated in the literature. Since streaming PCA models are not scalable in terms of dimensionality d and the number of eigenvectors k due to the orthogonalization processes, we also compare the performance with the partitioning-based schemes proposed in the previous chapters.

6.3 Analogy to The Wave Equation and Normal Mode Analysis

In this section, we show how well-known physical models are directly related to PCA. We first derive the PCA eigenvalue equation straight from the wave equation. We further show the consistency with the theory of classical mechanics. The implication of such consistency is that for a wide range of physical phenomena, one can automatically obtain the analytic model of a sequence of observations.

6.3.1 PCA Eigenvalue Problem from The Wave Equation

Given time-varying data of centered time-steps $X = [x_{t_1}, x_{t_2}, \dots, x_{t_n}] \in \mathbb{R}^{d \times n}$ where d is the total number of attributes, we make the following formulation. Each attribute corresponds to a spring state in a system of interconnected point masses. We assume that the points have equal masses but the setting in which point masses are interconnected is unknown. Since samples are centered, a negative attribute value indicates a compressed spring and a positive value indicates a stretched one. Hence, the system is described in terms of springs states rather than points positions. Since spring dynamics are analogous to the wave equation as discussed in Sec. 2.6.3, the time dependent behaviour of each spring can be expressed using the following partial differential equation

$$\frac{\partial^2}{\partial t^2} A - c^2 H A = 0, \tag{6.1}$$

where $A(x_0; t) \in \mathbb{R}^d$ is a vector function describing the state of each spring at time t and $x_0 \in \mathbb{R}^d$ are some initial conditions and H is the hessian matrix of all possible second order partial derivatives. Here, the Hessian matrix is used instead of the Laplacian operator in order to describe spring states in terms of the principal force direction. Note that in Sec. 2.6.3 example, the principal force direction was the X axis. By applying separation of variables and considering the eigenmode assumption, one can write the solution as follows

$$A(x; t) = V(x)\, T(t) = V e^{-iwt}, \tag{6.2}$$

where w is the eigenfrequency. This decomposes the eigenmode A into spatial $V(x)$ and temporal $T(t)$ components. The general solution of the wave equation(6.1) is then represented as a series of eigenmodes $\sum_i A_i$ of different eigenfrequencies. Each time-step can be reconstructed using the sum of all solutions $x = \sum_i A_i$. However in practice, time-steps can be approximated using only few eigenmodes that maximize the energy function $E = \mathbf{E}(A^T A) \leq \mathbf{E}(x^T x)$. Such maximization means that the Hessian matrix is negative semidefinite analogous to the second derivative test. It is well-known that the covariance matrix can be represented using the inverse of the hessian matrix in case of positive semi-definite Hessian for local minima A [GK04]. However, in case of negative semidefinite Hessian, the covariance matrix is represented using the inverse of the negative hessian matrix. This is due to the fact that the covariance matrix is always positive semi-definite. Hence, for the problem on hand, the covariance matrix can be approximated using the negative inverse of the Hessian matrix as follows

$$C = -H^{-1}. \tag{6.3}$$

In addition, form eq. (6.2) we get

$$\frac{\partial^2}{\partial t^2} A = w^2 A. \tag{6.4}$$

By plugging these into eq. 6.1 we get

$$\left(C - \left(\frac{c}{w}\right)^2 I\right) V e^{-iwt} = 0$$

$$\Rightarrow \left(C - \left(\frac{c}{w}\right)^2 I\right) V = 0 \tag{6.5}$$

which is precisely the PCA eigenvalue problem with λ being equal to $\left(\frac{c}{w}\right)^2$. Hence, we can visualize the first eigenmodes by simply projecting the time-steps onto the first significant eigenvectors and then weight them using the resulting projection value $y_i = A_i A_i^T x_t$. It can be noted that the higher significance eigenvectors will have lower eigenmode frequencies, which is consistent with the literature and the empirical study of this chapter.

6.3.2 A Classical Mechanics Interpretation

From the classical mechanics perspective, one can derive the PCA eigenvalue problem from the associated spring-mass system using the equation of motion. Since the restoring forces acting on a system of spring-mass are conservative, the force vector corresponds to the negative of the first derivative of the potential energy function, known as potential well, as follows

$$F = -\nabla \mathcal{V}(x), \tag{6.6}$$

where x is the spatial state of springs and $\mathcal{V}$ is the potential energy. But from Newton's second law of motion, we have

$$F = m \frac{\partial^2}{\partial t^2} x, \tag{6.7}$$

which yields

$$m \frac{\partial^2}{\partial t^2} x = -\nabla \mathcal{V}(x). \tag{6.8}$$

The potential energy around the equilibrium state can be approximated using Taylor series as follows

$$\mathcal{V}(x) = \mathcal{V}(x_{eq}) + \sum_{i=1}^{d} \left(\frac{\partial \mathcal{V}}{\partial x^i}\right)_{x_{eq}} x^i + \frac{1}{2} \sum_{i,j=1}^{d} \left(\frac{\partial^2 \mathcal{V}}{\partial x^i \partial x^j}\right)_{x_{eq}} x^i x^j + \cdots . \tag{6.9}$$

One can eliminate the first two terms since the potential energy is always zero at the equilibrium state and considering that the first derivative at the global minima is also zero. By neglecting higher order terms, this leads to the following matrix form of the potential energy

$$\mathcal{V}(x) = \frac{1}{2}x^T Hx. \tag{6.10}$$

Plugging eq. (6.10) into (6.8) yields

$$m\frac{\partial^2}{\partial t^2}A + HA = 0. \tag{6.11}$$

Here, we replaced the vector x by A since the solution to this eigenvalue problem gives the eigenmode (normal mode) at particular frequency (energy level) where the time step x is represented as the sum of these solutions $(x = \sum_i A_i)$. According to [AK01] the Hessian matrix of the potential energy is equivalent to

$$H = -kTC^{-1}, \tag{6.12}$$

where k is the spring constant and T is the temperature and the negative sign is due to the fact that the Hessian matrix is taken at local maximas of the potential energy function. Substituting eq. (6.12) in (6.11) and considering that $\frac{\partial^2}{\partial t^2}A = w^2 A$ give

$$\left(C - \left(\frac{kT}{mw^2}\right) I\right) A = 0, \tag{6.13}$$

which also corresponds to the PCA eigenvalue equation with $\lambda = \left(\frac{kT}{mw^2}\right)$.

6.3.3 Consistency with The Literature

As mentioned earlier, the analogy between PCA and NMA has been noted by many researchers especially in the area of Molecular Dynamics where PCA has been used for extracting the essential dynamics of proteins motion as an alternative to NMA. Having that said, we found that many research papers use a slightly different methodology when dealing with MD trajectory data. Recall that in MD trajectory data, each time-step represent the positions of m atoms in the Cartesian coordinate space (hence $n = 3m$). Many papers use a mass-weighted covariance matrix where the diagonal elements are weighted by the mass of the corresponding atom [Bas+98a; Bas+98b; MLS09; SMG00]. The main reason of using the mass-weighted covariance matrix is due to the fact that each atom in the actual molecule has its own mass (which may be different from other atoms) and according to Newton's Second Law, the force depends on both mass and acceleration. This implies that $F = M\frac{\partial^2}{\partial t^2}x$ where M is a diagonal matrix with each diagonal element being equal to the mass of the corresponding atom. However, in our formulation, we assumed the points in the associated system have equal masses and hence the mass matrix M is reduced to a scalar value that is included in the eigenvalue formula $\lambda = \left(\frac{kT}{mw^2}\right)$. Furthermore, according to [DA12], using the mass-weighted covariance matrix is very similar to NMA in that it is limited to only finding harmonic behaviour. On the other hand, using the standard covariance matrix can successfully extract both types of dynamics (harmonic and anharmonic) and hence is advantageous. This approach is known as essential dynamics and is one of the most widely used techniques for analyzing MD trajectories.

6.3.4 Case Study: Analyzing Dynamics of a Turbulent Vortex

As a proof of concept, we apply PCA for analyzing the dynamics of turbulent vortex observations from [Sil]. The data is of the form of time-varying volumes with 99 frames and $128 \times 128 \times 128$ voxels per frame, which is different from the type of dataset in molecular dynamics simulations. In this case, each voxel corresponds to a spring state in the associated spring-mass physical model. Figure 6.1 shows the projection values of time-steps on the first nine eigenvectors with their frequency spectrum. We can see at a glance that the scores of time steps have sinusoidal shaped curves which correspond to the time-dependent function $x_t^T V = T(t)$. One can also note from the spectrum that most significant eigenvectors have lower frequency with a higher amplitude which is consistent with our theoretical formulation. Figure 6.2 (a) and (b) show several frames from the vortex dataset compared to their reconstructions when using 20 eigenvectors where high similarity can be noted between the two sets. Figure 6.2 (c) shows the first 20 eigenvectors of the observations. It is evident that lower significance eigenvectors have higher spatial frequencies. In addition, the distributions of attributes in the eigenvolumes are not uniform and depend mostly on the observational data. Figure 6.2 (d) provides the explained variance curve, which shows that more than 95% of variations in dynamics are explained by only the first 20 eigenvectors.

In summary, the dominant behaviour of the system can be determined using the first few eigenvectors and their corresponding time-dependent functions. Each eigenvector has positive and negative attributes that define its vibration mode around the equilibrium state. The frequency of such vibration is reflected in the time-dependent function corresponding to projection values of time-steps onto this eigenvector. In general, higher significant eigenvectors have lower eigenfrequencies.

6.4 Empirical Evaluation

In this section, we compare the performance of streaming PCA methods and random-mapping Band-based PCA using several types of time-varying systems. We first test these methods for finding essential dynamics of large MD simulation, where applying the standard approach is not possible due to the large scale of the dataset. After that, we analyze daily weather readings of distributed weather stations around the globe from 1980 to 2016. Lastly, we study different physical phenomena in the form of time-varying volumetric data and video

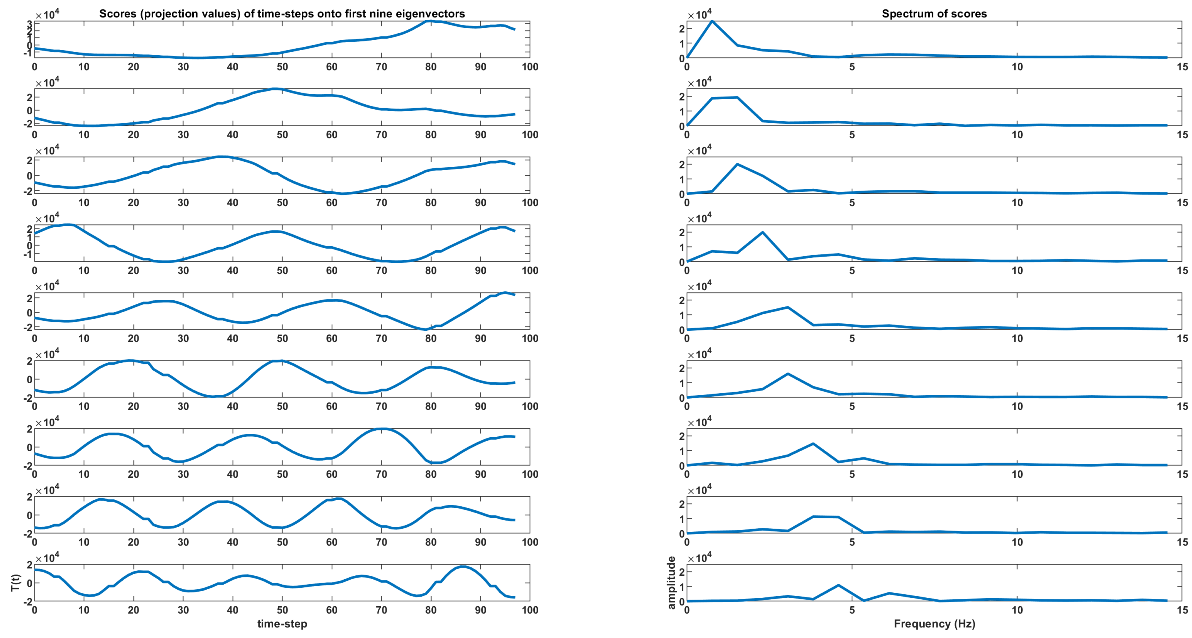


Figure 6.1: Projection values of time-steps onto the first nine eigenvectors (in descending order from top to bottom) and their frequency spectrum.

image series. Typically for such time-varying datasets, one would like the streaming PCA to converge rapidly to the actual normal modes of the simulation in an interactive manner. In other words, consider a physical simulation of a particular phenomenon, by applying streaming PCA, we should be able to interactively visualize the actual normal modes up to the recent time-step.

6.4.1 Learning Essential Dynamics of a Large MD Simulation

We first investigate the performance of streaming PCA and random-mapping BPCA when applied to a large-scale molecular dynamics trajectory. The dataset used in our study comprises reactions in a bacterial cell membrane obtained from [Gio+17]. The molecule contains $50,796$ atoms with 50,000 time-steps of trajectory. A visualization of the studied molecule in the equilibrium state is shown in figure 6.3 (a). Each time-step is represented using the Cartesian coordinates of each atom in the molecule. Therefore, a time-step will contain $3n = 152,388$ scalar values. Computing the eigenspace of such large-scale data using the standard approach is infeasible. Therefore, we evaluate the convergence of streaming PCA methods using MSE and explained variance. Table 6.1 contains the average MSE and percentage of explained variance when computing the first 30 eigenvectors using streaming PCA techniques. Unlike the case in time-independent datasets, the performance of SM, IPCA and CCIPCA was much better than the accelerated minibatch SGA. The SM algorithm in fully-online mode was giving the best convergence results. However, in comparison to its mini-batch variant in terms of processing speed, the full online mode took 1,770 seconds while the mini-batch approach took only 82 seconds with very similar convergence results. Visualizing the essential dynamics is done by superpositioning different modes of the first eigenvectors as shown figure 6.3 (b) where the first three essential dynamics produced using SM are visualized.

In order to compare performance between the holistic approach and random-mapping BPCA, figure 6.4 shows explained variance curves of SM streaming PCA and random-mapping BPCA. One can find that random-mapping BPCA gives better-explained variance curves for all band sizes. In general, the higher the band size, the more similar the explained variance curve to the SM curve. In addition, higher significance eigenvectors are more similar to the holistic solution than lower eigenvectors.

Table 6.1: Convergence of each method after a single data pass to the first 30 eigenvectors of the MD simulation in terms of MSE and explained variance.

Method	MSE	Explained Variance
mini-batch (block) power	1.174	22.1%
CCIPCA	0.392	74.4%
mini-batch CCIPCA	0.388	74.7%
IPCA	0.385	74.89%
SM	**0.3848**	**74.91%**
mini-batch SM	0.3853	74.88%
accelerated Oja (1st strategy)	0.4715	69%
accelerated Oja (2nd strategy)	0.4756	68.8%
accelerated block power (1st strategy)	0.4756	68.82%
accelerated block power (2nd strategy)	0.4716	69%

6.4.2 Analyzing Daily Weather Readings From 1980 to 2016

We now analyze daily weather readings from the Global Surface Summary of the Day database (NOAA GSOD). The database is created by the National Oceanic and Atmospheric Administration and is updated on a daily basis starting from 1929 with more than 20 weather attributes (average temperature, wind speed, etc) collected from over 9,000 stations around the globe. However, only a few stations have been operating since 1929. In addition, acquiring readings from the database is not easy since many stations do not provide readings regularly, not to mention that readings may contain missing values. In our experiments, we considered stations operating from 1st January 1980 to 31st December 2016. By operating we mean stations that provide at least three reading in a year from 1980 to 2016. In order to eliminate missing values we linear-interpolate the readings throughout the year. We created three datasets: Average temperature, average wind speed and see level pressure. The temperature and wind readings were acquired from 2,720 distributed stations and the pressure dataset is based on 1,794 stations readings. Each dataset contains 13,515 time-steps.

Figure 6.5 (a) shows the locations of the weather stations used for acquiring the readings.

Figure 6.5 (b, c and d) show the first three vibration modes of each weather dataset. By analyzing the time-dependent functions of the modes from 1980 to 1984 as illustrated in figure 6.6 we can observe a repetition in patterns over each year. This suggests that such dynamics correspond to seasonal changes in the weather. The wind dataset was shown to have a more stochastic nature compared to the other datasets. One can also note that lower eigenvectors produce noisy time curves.

In order to evaluate the convergence of streaming PCA to the optimal solution, we ran 10 trials for each technique and reported mean and standard deviation of the convergence to the standard PCA solution after a single data pass. For mini-batch methods, the mini-batch size was set to 100. Table 6.2 reports the convergence results of each streaming PCA algorithm to the top 10 eigenvectors. For all datasets, CCIPCA and IPCA are achieving the best results. The SM algorithm performance is not as good as in the MD simulation. For the accelerated mini-batch SGA, the first strategy generally converges better than the second one, unlike the case with non-time-varying datasets. This is well-reflected in figure 6.7 where the convergence rates based on a single trial are illustrated for each dataset. Figure 6.8 provides the explained variance curves of standard PCA and random-mapping BPCA for each dataset. For the temperature dataset, 83% of the dynamics are covered by only the first eigenvector and 94% are covered by the first first 30 eigenvectors. For the pressure dataset, the first 30 eigenvectors cover 89% of the variations. On the flip-side, for the wind dataset, the first 30 eigenvectors cover only 40%. In general, the explained variance of random-mapping BPCA is better than the standard approach. Consistent with the behaviour in MD simulation, the gap between random-mapping BPCA and the holistic approach in terms of the explained variance curves is very narrow especially for the first three eigenvectors and then it slightly increases as the number of eigenvectors increases. It is interesting to note that the gap in the wind dataset is larger than those in the temperature and pressure datasets. This might be a result of the stochastic pattern observed from the time-dependent function. However, investigating the main reason of such wider gaps is subject to further research.

6.4.3 Other Time-varying Datasets

We investigate other time-varying datasets of video and 4D volumetric formats, namely, simulation of ocean waves, supernova and vortex observations. Table 6.3 includes a summary of each dataset. Figure 6.9 shows several frames from the Supernova dataset reconstructed using first 15 eigenvectors where 92% of variability is covered. For the ocean waves simulation, almost 98% of dynamics are explained using only 10 eigenvectors. In general, one can note from the visualizations that lower eigenvectors have higher spatial frequencies. Such high frequencies correspond to an increased level of detail in reconstructions.

Similar to what we did in the previous subsection, we evaluate the performance of each method

Table 6.2: Convergence of each method after a single data pass to the first ten eigenvectors of the weather datasets.

Method	Temperature	Wind	Pressure
mini-batch (block) power	-1.4 ± 0.01	-0.346 ± 006	-0.93 ± 0.01
CCIPCA	-3.24 ± 0.07	-1.75 ± 0.05	-2.3 ± 0.13
mini-batch CCIPCA	$\mathbf{-3.4 \pm 0.1}$	$\mathbf{-1.9 \pm 0.09}$	$\mathbf{-2.5 \pm 0.2}$
IPCA	-3.37 ± 0	-1.79 ± 0	-2.36 ± 0
SM	-1.85 ± 0.3	-1.87 ± 0.11	-1.96 ± 0.4
mini-batch SM	-2.48 ± 0.39	-1.8 ± 0.15	-2.4 ± 0.28
accelerated Oja (1^{st} strategy)	-2.87 ± 0.08	-1.67 ± 0.06	-2.3 ± 0.22
accelerated Oja (2^{nd} strategy)	-2.53 ± 0.02	-1.26 ± 0.02	-2.1 ± 0.02
accelerated block power (1^{st} strategy)	-2.85 ± 0.09	-1.67 ± 0.04	-2.37 ± 0.2
accelerated block power (2^{nd} strategy)	-2.53 ± 0.02	-1.26 ± 0.03	-2.07 ± 0.06

by running 10 trails for each technique and then reporting mean and standard deviation of the convergence after a single data pass. For all mini-batch methods, we set the block size to 5. Table 6.4 shows the convergence achieved by each method after processing all times-steps. It can be noted that CCIPCA (in both online and minibatch modes) and IPCA are achieving the best convergence results especially for the ocean waves and Supernova simulations. However, the CCIPCA is superior to IPCA in that it can work in both online and mini-batch modes. For our acceleration, the 1^{st} strategy is converging better than the second one. Overall, one can conclude from all experiments in this section that CCIPCA, in both online and mini-batch settings, always achieves good convergence results with robust performance. Hence, for time-varying datasets, CCIPCA is preferable amongst other streaming PCA approaches. For random-mapping BPCA, we visually inspect the analogy between eigenvectors produced using random-mapping and the standard holistic approach as shown in figure 6.11. The band size was set to 1,000 for wave simulation and 3,000 for the supernova and vortex datasets. We can see that the random-mapping BPCA eigenvectors resemble the holistic ones but with some dithering effects. This is consistent with what was found for non-time-varying datasets in the previous chapters. Hence, for large-scale time-varying datasets, the random-mapping BPCA can serve as a more scalable approach for approximating the holistic eigenvectors. Such scalability makes random-mapping BPCA even superior over the holistic streaming approaches.

6.4.4 Comparing Reconstruction Quality with CPCA and mean-variance BPCA

We observed from the previous subsections that the eigenvectors computed using random-mapping BPCA are related to the holistic solution. We now discuss the performance of CPCA and mean-variance BPCA for time-varying data. Figure 6.12 compares explained variance curves of holistic PCA and the two partitioning approaches for the waves, supernova and vortex datasets. For CPCA, the cell-size was set to $10 \times 10 \times 10$ for supernova and vortex datasets and to 10^2 for the wave dataset. For BPCA, the band-size was set to 100 for the wave dataset and to $1,000$ for the other datasets. It is clear that for all datasets CPCA is giving the best results where only four eigenvectors are sufficient to explain more than 90% of the variability. The mean-variance BPCA falls between CPCA and the holistic solution. Its performance for the waves and supernova datasets was much better than the vortex dataset where its explained variance curve was very close to the holistic approach. The use of CPCA or mean-variance BPCA may not provide any physical interpretation of the system. In other words, visualizing the top eigenvectors may not correspond to the actual vibration modes of the analytic solution. However, the main advantage that one would get from such approaches is the higher compression ratio gained. For instance, the explained variance achieved using only 4 cell eigenvectors for the turbulent vortex dataset is equivalent to the explained variance acquired using 14 holistic eigenvectors. Since the compression ratio for the holistic approach corresponds to $\frac{nd}{k(n+d)}$ and for the CPCA the compression ratio equals $\frac{nd}{k(nd/d_{cell}+d)}$ where d_{cell} is the number of attributes (voxels) per cell. This means that the compression ration for CPCA is 22:1 and for standard PCA the compression ratio is only 7:1 when maintaining the same explained variance of 90%.

Table 6.3: Summary of the time-varying simulations used in the experiments.

	Ocean waves	**Supernova**	**Turbulent Vortex**
no. of samples	300	60	99
sample type	100×100 images	$400 \times 400 \times 400$ volumes	$125 \times 125 \times 125$ volumes
Reference	[Nvi]	[**supernova**]	[Sil]

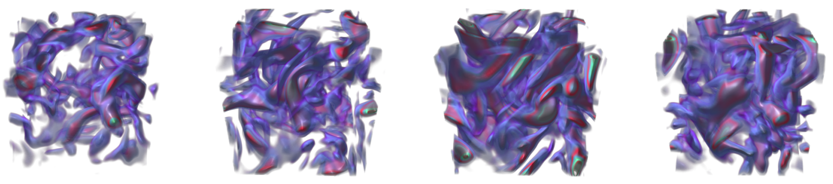
Table 6.4: Convergence of each method after a single data pass to the first five normal modes of each simulation.

Method	Ocean Waves	Supernova	Vortex
mini-batch (block) power	-0.6379 ± 0	-0.21 ± 0	-0.096403 ± 0
CCIPCA	$\mathbf{-2.4521\pm 0.021}$	$\mathbf{-1.74\pm 0.04}$	$\mathbf{-1.1773\pm 0}$
mini-batch CCIPCA	$\mathbf{-2.7\pm 0.18}$	$\mathbf{-1.4\pm 0.04}$	$\mathbf{-1.01\pm 0.22}$
IPCA	$\mathbf{-2.47\pm 0.011}$	$\mathbf{-1.76\pm 0}$	$\mathbf{-1.0\pm 0}$
SM	-1.73 ± 0.059	-0.59 ± 0.01	-0.45 ± 0.13
mini-batch SM	-1.5 ± 0.2	-0.57 ± 0.006	-0.79 ± 0.1
accelerated Oja (1^{st} strategy)	-1.52 ± 0.0259	-0.88 ± 0.058	-0.75 ± 0.06
accelerated Oja (2^{st} strategy)	-1.46 ± 0.046	-0.54 ± 0.095	-0.4 ± 0.067
accelerated block power (1^{st} strategy)	-1.58 ± 0.09	-0.97 ± 0.076	-0.77 ± 0.11
accelerated block power (2^{st} strategy)	-1.49 ± 0.12	-0.52 ± 0.1	-0.42 ± 0.05

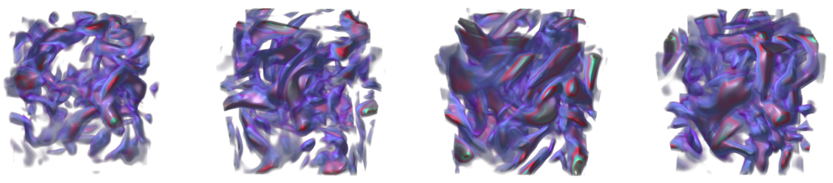
6.5 Conclusions

In this chapter, we employed PCA for analyzing time-varying datasets. Particularly, we addressed physical time-varying systems that are based on the wave equation. Such a partial differential equation has been immensely used for modelling a broad scope of physical phenomena. We showed that the eigenvectors obtained using PCA for such type of data are directly related to the analytic model of the underlying physical phenomenon. Basically, PCA decomposes the dynamics of such systems into a number of vibration modes (also known as normal-modes, eigenmodes or standing waves) in which most of the system dynamics are ruled by only a small number of such modes. Since time-varying data acquired by scientific simulations or physical observations are usually very large in scale, computing the domain dynamics using the standard PCA approach may not be feasible. We proposed using streaming PCA to remedy such a problem. To the best of our knowledge, the efficiency of streaming PCA for such domain application has not been investigated in the literature. By analyzing the convergence results of streaming PCA algorithms on different time-varying datasets, we found that CCIPCA and IPCA provide very good steady performance for all of the studied datasets. However, CCIPCA is more practical compared to IPCA since it can be applied in both online and mini-batch modes. The main limitation of streaming PCA is its quadratic computation dependence on the number of eigenvectors k. In practice, the appropriate number of eigenvectors for describing a time-varying model may be very large due to the gap between highest and lowest frequencies in the vibrating modes such as the case in protein dynamics. This raises the need for more scalable paradigms. Therefore, we further examined the analogy between random-mapping BPCA and the standard holistic approach. While there was a clear resemblance between the two models in terms of first eigenvectors, the explained variance of random-mapping PCA was surprisingly better than standard PCA. In general, we noticed that random-mapping BPCA produces a dithered version of the holistic eigenvector. Lastly, we showed that using CPCA instead of random-mapping significantly enhanced the explained variance. Such enhancement suggests that CPCA may be a more practical method for compressing such large-scale data. Investigating other mapping strategies for BPCA that would outperform CPCA in terms of reconstruction quality would be an interesting topic for future research. This is of paramount significance for compressing non-spatially localized time-varying datasets.

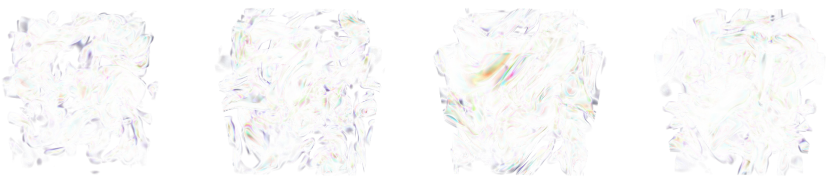
(a)



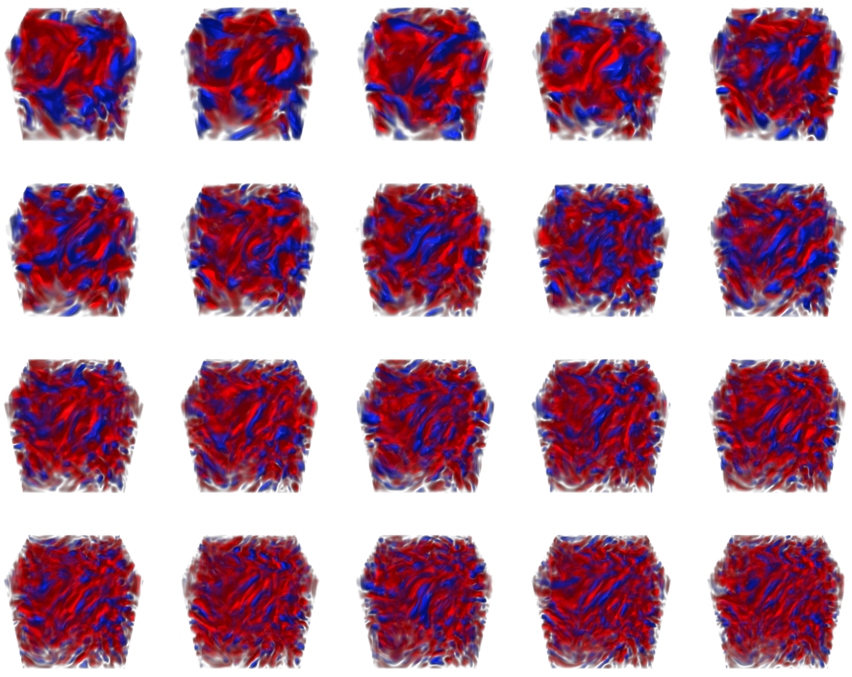
(b)



(c)



(d)



(e)

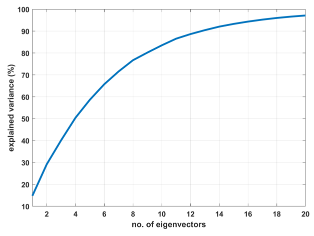
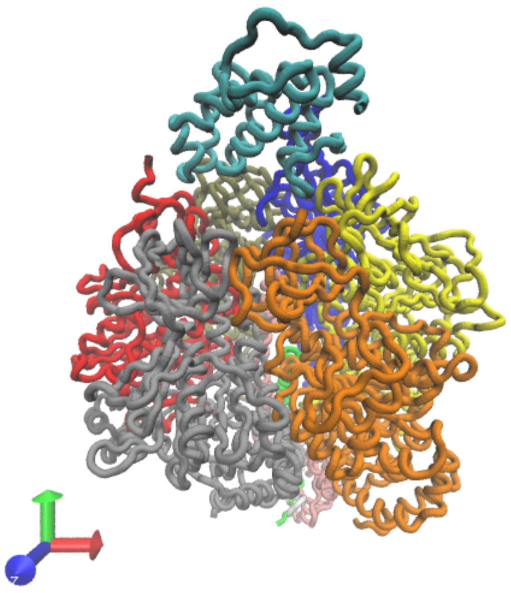


Figure 6.2: Learning dynamicsin a turbulent vortex using PCA. *(a)* Original frames from the dataset. *(b)* Reconstructed frames using 20 eigenvectors. *(c)* Difference image. *(d)* First 20 eigenvectors in descending order from left to right and from top to bottom where red areas correspond to negative attributes and blue areas correspond to positive ones. *(e)* Explained variance curve of the first 20 eigenvectors.

(a)



(b)

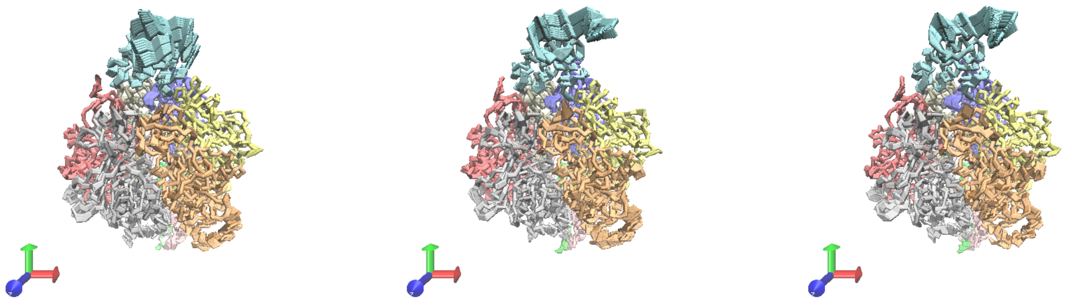


Figure 6.3: Visualization of Essential Dynamics. *(a)*Average molecule in ribbon representation colored based on segment name using VMD software. *(b)* Superposition of the first three essential dynamics computed using similarity matching.

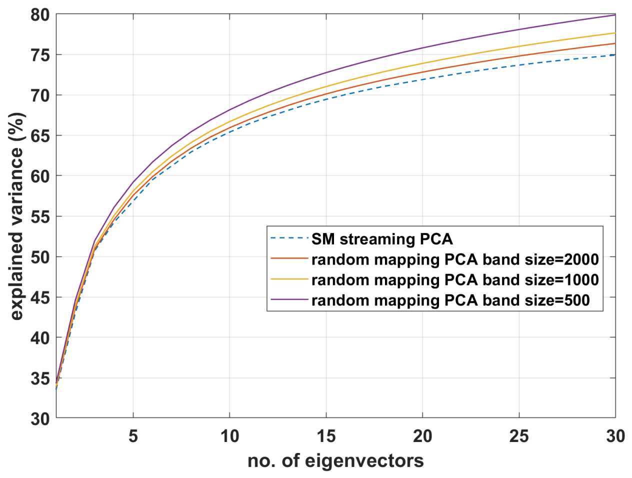


Figure 6.4: Comparing performance between SM streaming PCA and random-mapping BPCA using explained variance curve.

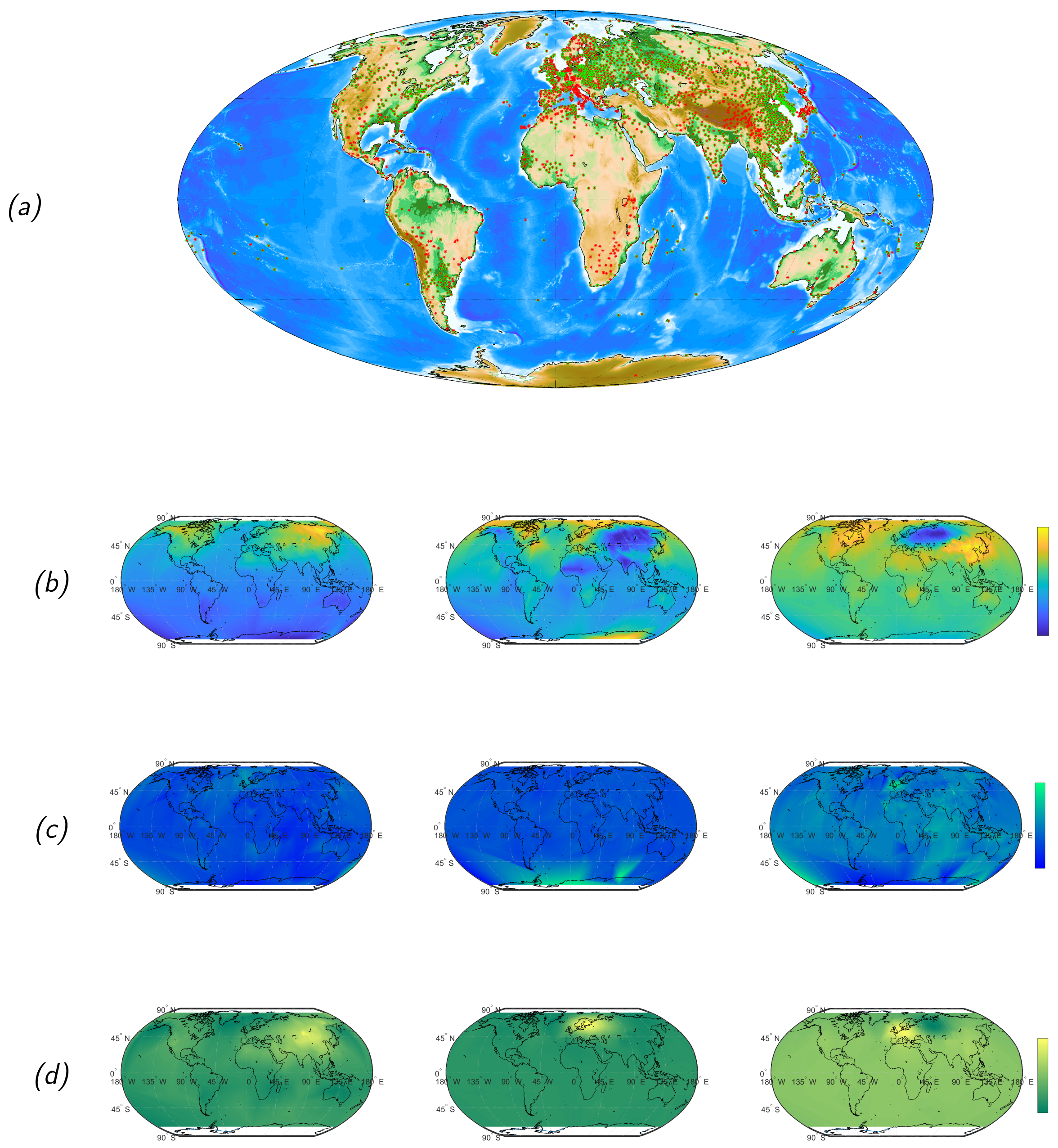


Figure 6.5: Visualizing the dominant vibration modes (eigenmodes) of each weather dataset. *(a)* Locations of weather stations used for acquiring readings where red stars indicate temperature and wind stations and green circles indicate pressure stations. *(b)* The top three vibration modes of the temperature dataset. *(c)* The top three vibration modes of the wind dataset. *(d)* The top three vibration modes of the pressure dataset.

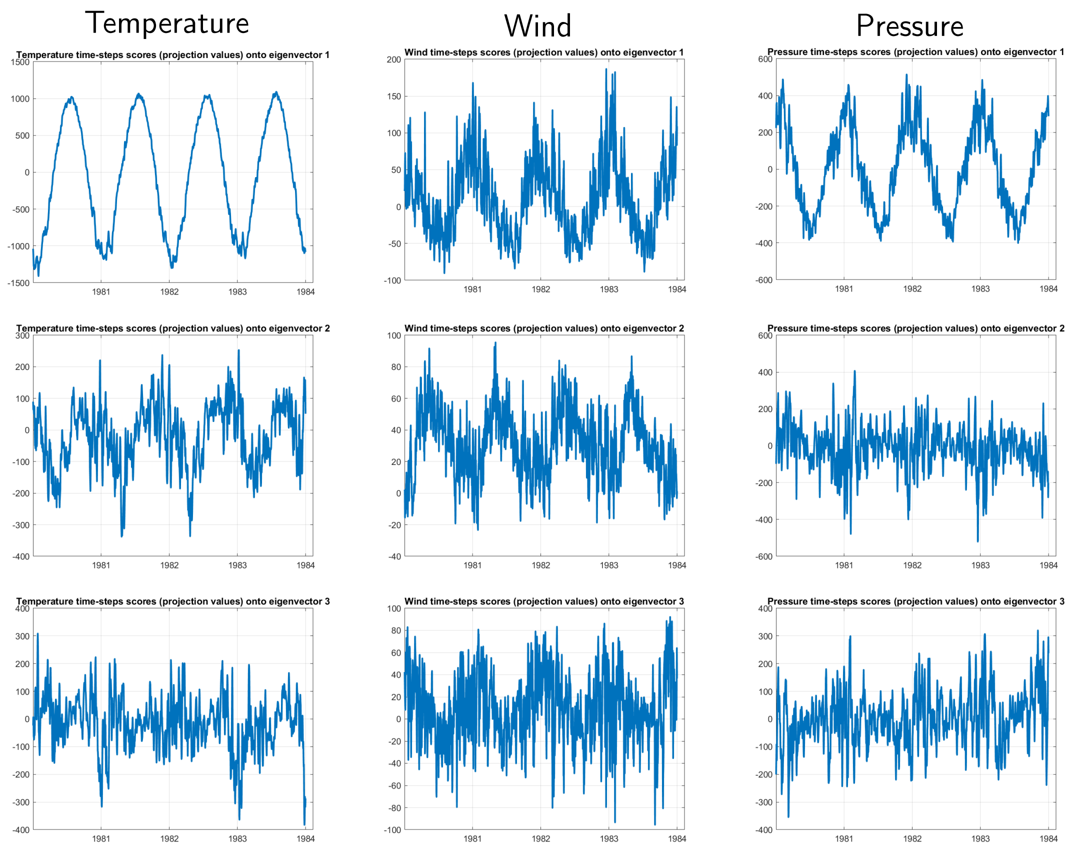


Figure 6.6: Time-dependent functions of the first three vibration modes of each weather dataset.

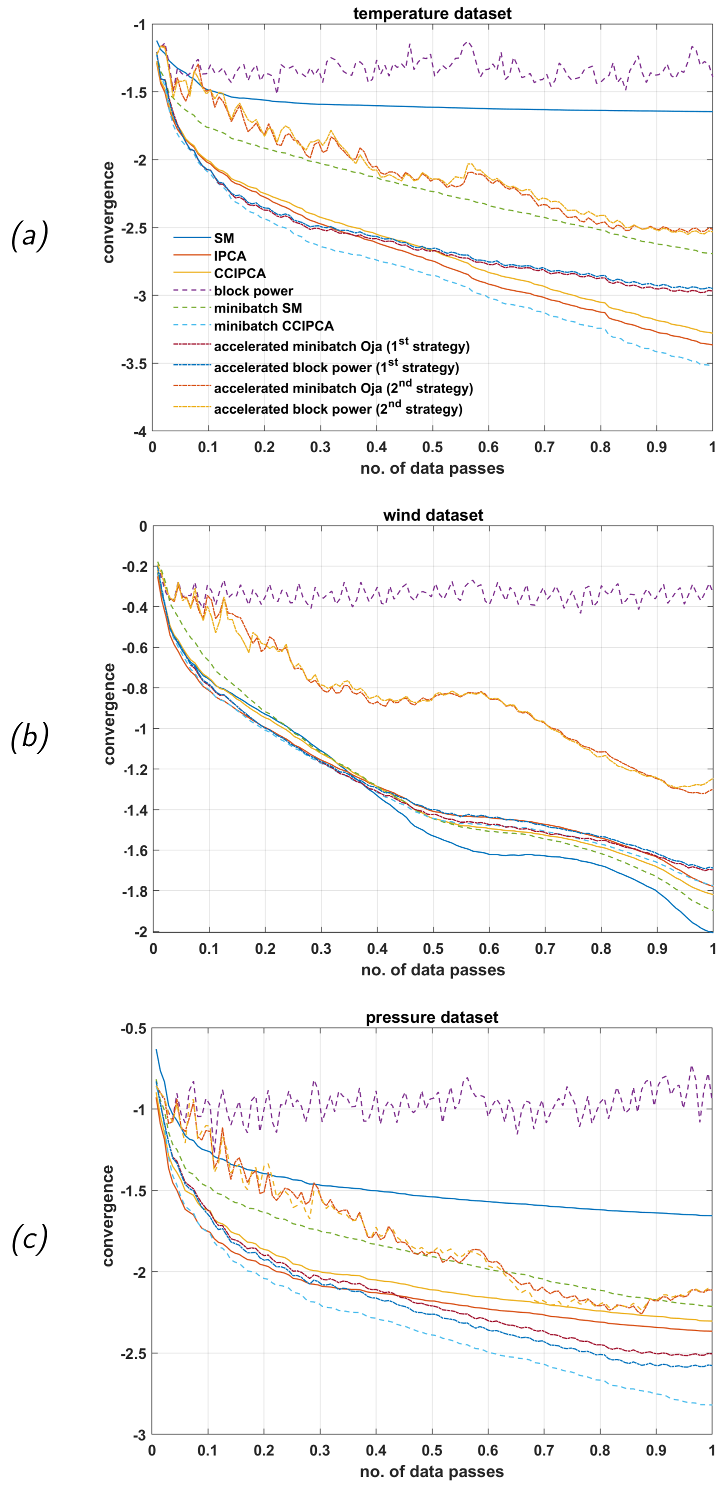
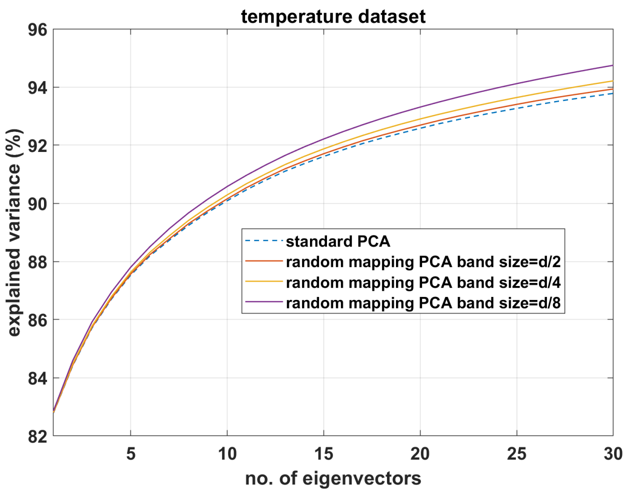
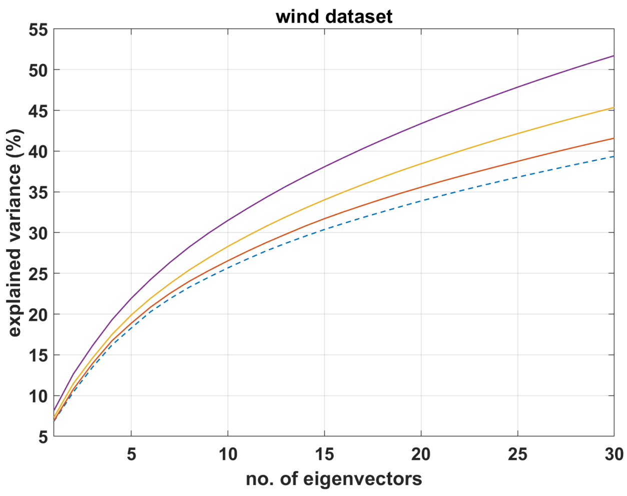


Figure 6.7: Convergence rates to the top 10 eigenvectors based on a single-trial for (a) temperature, (b) wind and (c) pressure datasets.

(a)



(b)



(c)

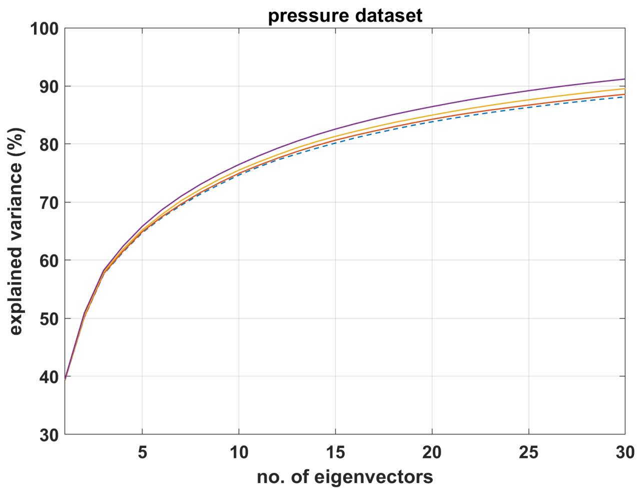


Figure 6.8: Explained variance curves of standard PCA and random-mapping BPCA for *(a)* temperature, *(b)* wind and *(c)* pressure datasets.

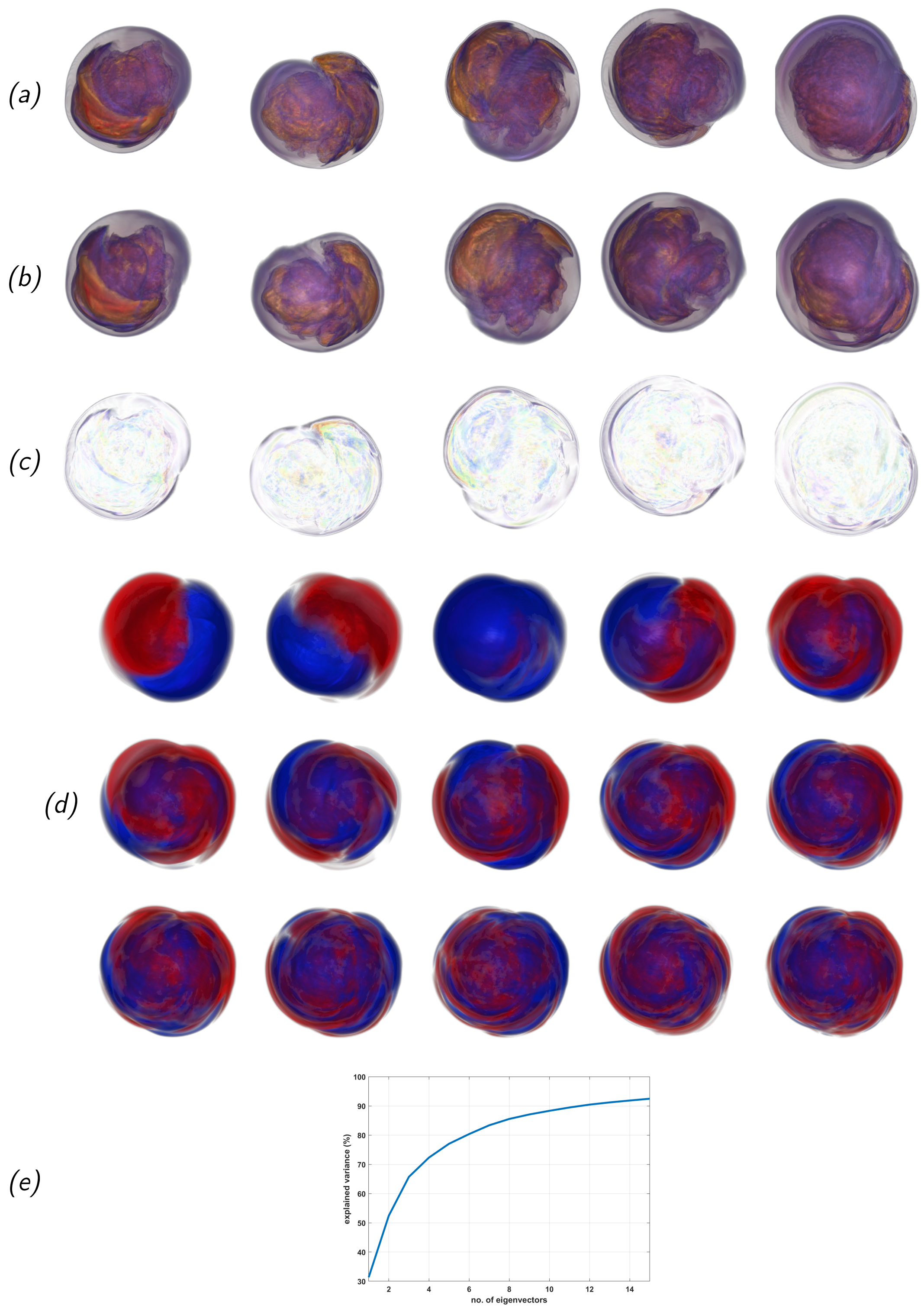


Figure 6.9: Learning dynamics from Supernova dataset. *(a)* Original frames from the dataset. *(b)* Reconstructed frames using 15 eigenvectors. *(c)* Difference image. *(d)* First 15 eigenvectors in descending order from left to right and from top to bottom where red areas correspond to negative attributes and blue areas correspond to positive ones. *(e)* Explained variance curve of the first 15 eigenvectors.

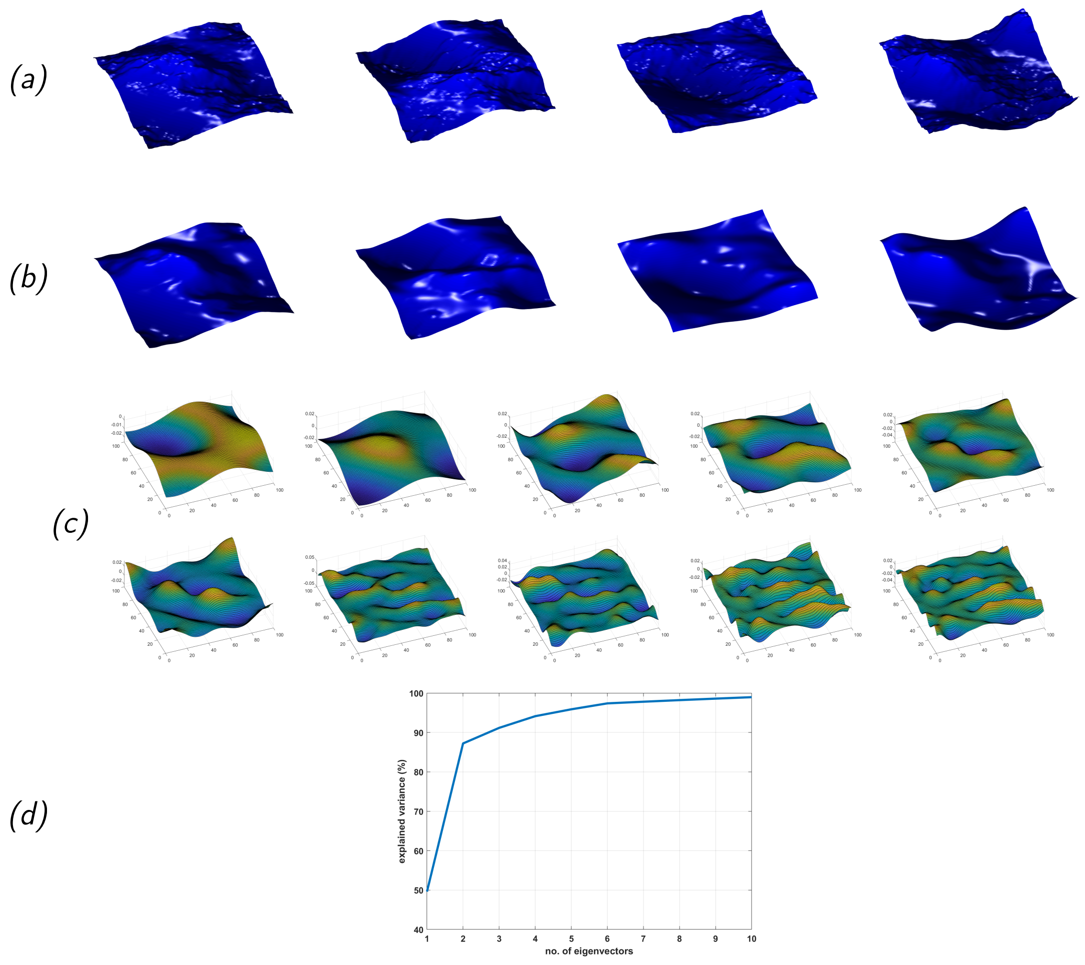


Figure 6.10: Learning dynamics from ocean waves simulation. *(a)* Original frames from the dataset. *(b)* Reconstructed frames using 10 eigenvectors. *(c)* First 10 eigenvectors in descending order from left to right and from top to bottom. *(d)* Explained variance curve of the first 10 eigenvectors.

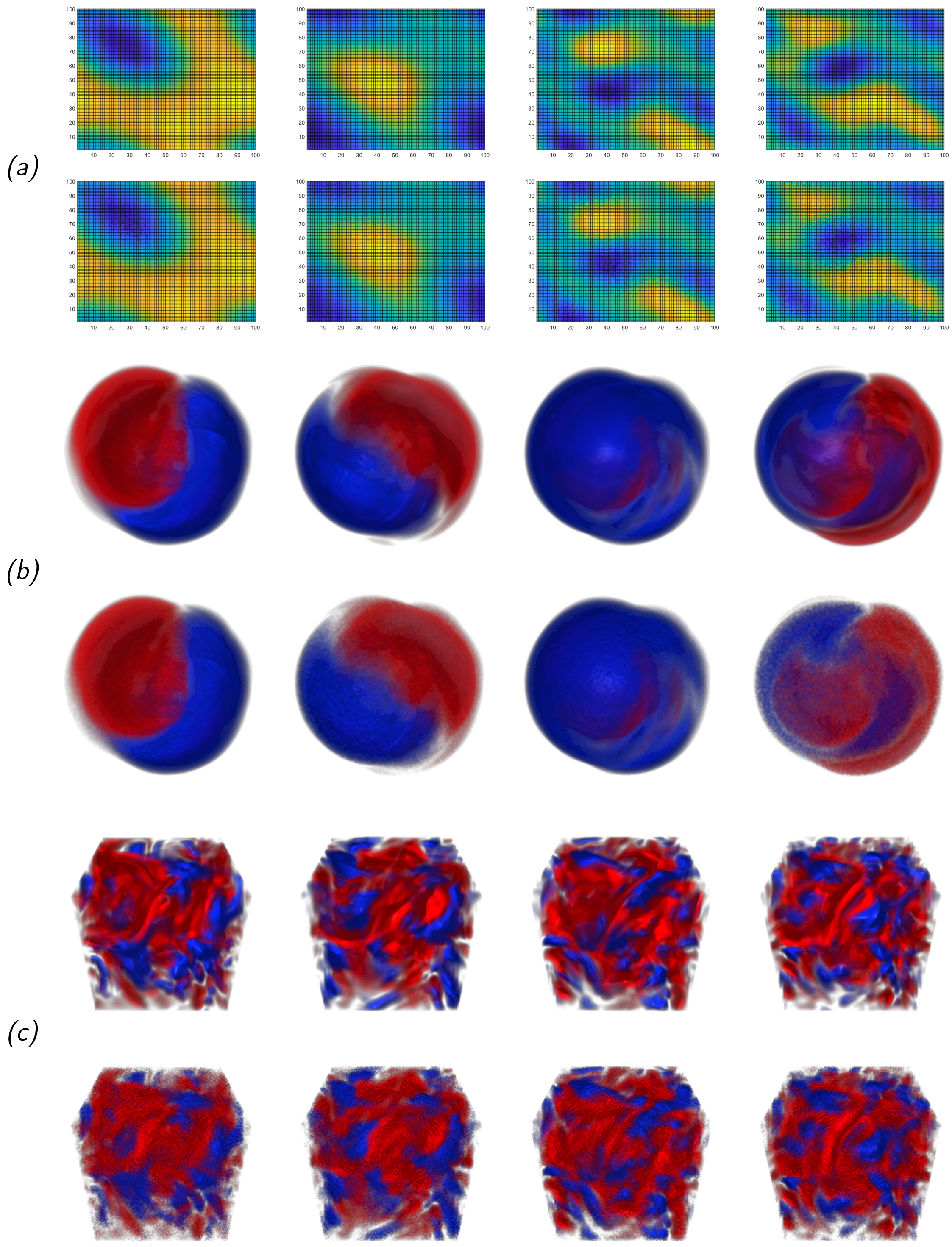


Figure 6.11: Visual comparison between random-mapping BPCA eigenvectors and holistic ones for *(a)* ocean waves simulation *(b)* Supernova and *(c)* turbulent vortex datasets.

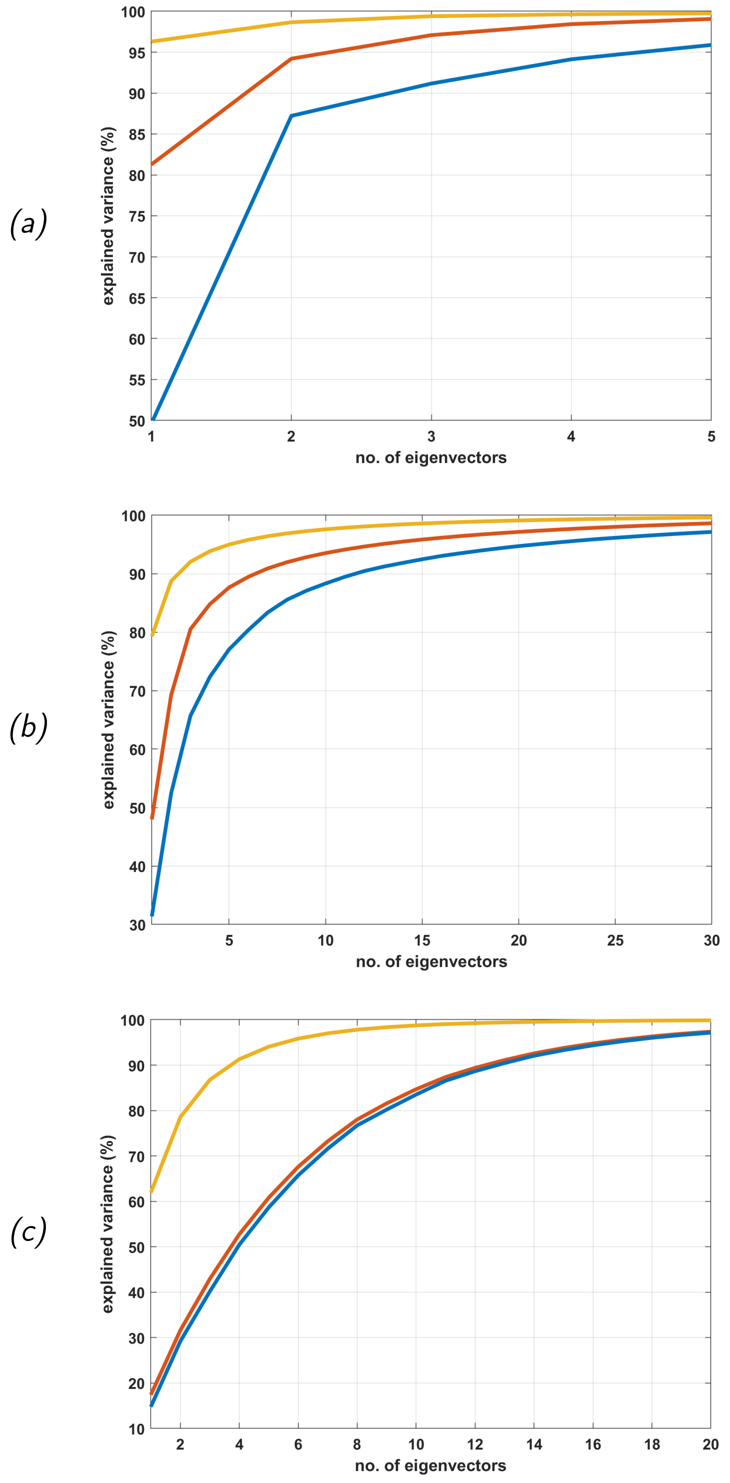


Figure 6.12: Explained variance curves of CPCA (yellow curve) and mean-variance BPCA (red curve) compared to holistic PCA (blue curve) for *(a)* ocean waves simulation *(b)* Supernova and *(c)* turbulent vortex.

7 Conclusions and Future Work

7.1 Summary of Contributions

In this thesis, we investigated effective strategies for modelling large-scale datasets. By modelling, we mean extracting meaningful patterns that describe most variations in the input dataset. PCA is an unsupervised learning algorithm that has widely been used for such a purpose due to the rather solid theoretical ground that it enjoys. However, the standard approach to PCA involves an eigenvalue-decomposition of the covariance matrix, which is computationally demanding with quadratic complexity dependence on the data size. This is particularly problematic when dealing with large-scale datasets. Designing a reduced complexity PCA for such datasets has been widely explored in the literature. Ideally, the eigenvectors should be computed from a single data pass where samples are visited only once.

A review of related reduced-complexity techniques was provided in Chapter 2. One limitation in current state-of-the-art treatments is due to their sequential nature. This, in turn, significantly limits the run-time which is problematic when considering time-critical applications. In addition, most well-known techniques that are based on the stochastic gradient approximation involve a learning rate parameter for which an intuitive choice is not applicable. Such a learning rate sensitivity may result in a very slow convergence rate. In fact, many studies showed that the optimal choice of learning rate for SGA methods necessitates knowing the eigengap, the computation of which requires a pre-processing data pass violating the main condition of online learning. Another important phenomenon that may occur with streaming schemes is that when computing multiple eigenvectors, the produced eigenvectors may not resemble the real ones but rather converge to a span of the original eigenspace. We also visited other related topics in order to introduce some concepts that are used in developing our techniques. While many modern studies focus on the theoretical analysis, in most of this thesis, we followed empirical evaluation strategies.

Mini-batching is a simple yet effective strategy for speeding up computation. In the PCA literature, this technique has been applied to the power iterations method where the update-step is applied on mini-batches of the dataset that are visited only once. However, a satisfactory convergence demands a large mini-batch size which is prohibitive in the case of large-scale datasets. In Chapter 3, an acceleration scheme for mini-batch SGA methods was proposed. The approach is based on the fact that the optimal solution corresponds to a steady-state where successive updates do not change. Hence, our acceleration catalyzes the convergence of the original methods by seeking such a state. We also studied the performance of other state-of-the-art methods when extended to the mini-batch mode. We evaluated the performance using the spiked covariance model and other benchmark datasets, including ones whose eigenvectors cannot be computed using the standard PCA due to their high dimensionality. For such datasets, we evaluated the convergence using the mean-squared error. Using relatively smaller batch sizes compared to the literature, our acceleration outperformed the original methods as well as other state-of-the-art approaches. In addition, the computation time was significantly reduced compared to the fully-online mode. We also employed our scheme for initializing the batch power iteration and showed how such an initialization significantly improved the convergence results. Having that said, a detailed theoretical analysis of the rate of convergence is lacking. Moreover, this scheme is limited when considering very high dimensional data where even a single sample may not fit into a typical machine memory space.

A tacit reason for the sequential nature of streaming PCA methods is that PCA is usually considered as a holistic representation. In Chapter 4, we investigated the merits of applying PCA in a partitioning manner. One main inspiration was the JPEG compression standard where DCT is performed on sub-blocks of the input image. In the literature, sample-based partitioning was widely investigated where samples are mapped into different subsets based on some criterion, and then PCA is applied to each set individually which is known as subspace clustering. However, applying attribute-based partitioning was not widely explored. We proposed two models for attribute-based partitioning. The first model, Cell-based PCA (CPCA), subdivides each image into uniform blocks and then applies PCA to each block separately. Two main gains were observed. Firstly, the reconstruction error was significantly reduced in comparison to the holistic approach. Secondly, the technique is embarrassingly parallel, leading to remarkably faster computation. On the flip side, the cell-based model is restricted in that it can only deal with spatially structured datasets such as images and volumes. In addition, cell boundary artifacts can be noted when using an insufficient number of cell eigenvectors. The second partitioning model, Band-based PCA (BPCA), partitions attributes based on their values distribution rather than their spatial locations and hence can be generalized to non-spatially localized data. We found that mapping based on the mean and variance is better in terms of reconstruction quality compared to mean-only mapping. Moreover, estimating the mean and variance using larger population sizes enhanced the results which reflects the soundness of the technique. However, the reconstruction quality of CPCA was found to be better. Another important finding in this chapter is that the baseline model corresponding to random-mapping BPCA produces a solution that highly resembles the holistic eigenspace. This suggests that the random-mapping BPCA may be a more practical alternative to streaming PCA due to its parallel and scalable nature.

From another perspective, it is well-known that state-of-the-art deep learning models do implicitly perform dimensionality reduction and feature extraction but in a rather stochastic manner. However, the parametric and stochastic nature of such models makes it hard to arrive at an optimal (inner) representation. In addition, modern benchmark datasets dedicated to classification tasks provide a vast number of training samples for only a few classes. For instance, the CIFAR-10 dataset contains 50,000 training samples for only 10 classes. Chapter 5 utilized CPCA to derive simple and meaningful lightweight data representation that would improve the accuracy of deep learning models. Unlike the dimensionality reduction involved in deep CNN, this has the advantage of being deterministic. In addition, we designed a novel deep CNN classifier, RedNet, that can efficiently learn from limited and reduced-size examples. The model is inspired by the well-known AlexNet classifier but involves more cross channel normalization layers. We showed that the accuracy of RedNet was significantly improved when representing images using CPCA in comparison to the conventional bicubic downsampling. Experiments on LFW face dataset depicted how such a model outperforms state-of-the-art deep CNN classifiers when trained with limited examples.

Modelling time-varying phenomena is of high importance for understanding key aspects in different scientific fields. In fact, such systems appear naturally in our world. In Chapter 6, we applied PCA as an automatic modelling tool for a wide range of time-varying applications. We theoretically studied the analogy between well-known Physical models and PCA. In particular, we derived the PCA eigenvalue equation directly from the wave equation. In addition, we showed the consistency with the theory of classical mechanics. This suggests that the solution obtained using PCA for a sequence of time-varying physical observations can appropriately reflect its underlying analytic model. This is mainly due to the way PCA decomposes such observations into their vibrational modes. Since time-varying datasets are no exception from being massively large, it was important to investigate reduced-complexity PCA schemes for such a core application. To the best of our knowledge, the performance of streaming PCA methods for analyzing time-varying datasets has not been examined in the literature. By analyzing the convergence results of streaming PCA algorithms on different time-varying datasets, we found that CCIPCA, in both online and mini-batch modes, provides a very good performance for all of the studied datasets. The main limitation of streaming PCA is its quadratic computation dependence on the number of eigenvectors k. In practice, the appropriate number of eigenvectors for describing a time-varying model may be very large due to the gap between the highest and lowest frequencies in the vibrating modes such as the case in protein dynamics. This raises the need for more scalable paradigms. Therefore, we further examined the analogy between random-mapping BPCA and the standard holistic approach. While there was a clear resemblance between the two models in terms of the first eigenvectors, the explained variance of random-mapping BPCA was surprisingly better than the standard PCA. In general, we noticed that random mapping BPCA produces a dithered version of the holistic eigenvectors. Lastly, we showed that using CPCA instead of random-mapping significantly enhanced the explained variance curve. Such enhancement suggests that CPCA may be a more practical method for compressing such type of datasets.

In summary, this thesis provides practical solutions that allow for efficient modelling of various types of large-scale datasets. For holistic data representation, we recommend using our acceleration scheme in the case of independent and identically-distributed samples and CCIPCA in case of time-varying observations. For larger-scale datasets, the random-mapping BPCA can be employed. In order to achieve a better reconstruction quality and feature extraction, the cell and band-based models should be considered.

7.2 Future Research

In terms of future work, my thesis has opened up various areas that are worthy of further investigation. Firstly, it would be interesting to investigate whether our acceleration in Chapter 3 can be generalized for solving other SGA optimization problems. Secondly, Chapter 4 applied PCA for representing the individual partitions. Representing such partitions using other dimensionality reduction approaches such as Linear Discriminant Analysis is kept for future research. A further intuitive extension would be to study the performance of non-linear schemes such as kernel PCA when applied in the partitioning mode. Chapter 5 showed that using CPCA significantly enhanced the perception of deep CNN. An important question to raise is that: Can reducing the dimensionality solely using a cascade of CPCA operations be advantageous to the conventional CNN model? Last but not least, Chapter 6 used reduced complexity schemes for time-varying data. Designing a hardware optimized implementation for such schemes that can achieve real-time rates would be of significant importance to many time-critical applications.

Bibliography

[ANR74] Nasir Ahmed, T_ Natarajan, and Kamisetty R Rao. "Discrete cosine transform". In: *IEEE transactions on Computers* 100.1 (1974), pp. 90–93.

[Air+18] Sagar Airen et al. "Normal-mode analysis for collective neutrino oscillations". In: *Journal of Cosmology and Astroparticle Physics* 2018.12 (2018), p. 019.

[AD16a] Salaheddin Alakkari and John Dingliana. "Volume Rendering Using Principal Component Analysis". In: *EuroVis 2016 - Posters*. Ed. by Tobias Isenberg and Filip Sadlo. The Eurographics Association, 2016. ISBN: 978-3-03868-015-4.

[AD16b] Salaheddin Alakkari and John Dingliana. "Volume Rendering Using Principal Component Analysis." In: *EuroVis (Posters)*. 2016, pp. 85–87.

[AD18a] Salaheddin Alakkari and John Dingliana. "A Multi-View Image-Based Volume Visualization Technique". In: *IEEEVis (Posters)*. 2018.

[AD18b] Salaheddin Alakkari and John Dingliana. "Principal Component Analysis Techniques for Visualization of Volumetric Data". In: *Advances in Principal Component Analysis*. Springer, 2018, pp. 99–120.

[AGC15] Salaheddin Alakkari, Eugene Gath, and John James Collins. "An investigation into the use of subspace methods for face detection". In: *Neural Networks (IJCNN), 2015 International Joint Conference on*. IEEE. 2015, pp. 1–7.

[AL17] Zeyuan Allen-Zhu and Yuanzhi Li. "First efficient convergence for streaming k-pca: a global, gap-free, and near-optimal rate". In: *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2017, pp. 487–492.

[ALB93] Andrea Amadei, Antonius Linssen, and Herman JC Berendsen. "Essential dynamics of proteins". In: *Proteins: Structure, Function, and Bioinformatics* 17.4 (1993), pp. 412–425.

[AK01] Ioan Andricioaei and Martin Karplus. "On the calculation of entropy from covariance matrices of the atomic fluctuations". In: *The Journal of Chemical Physics* 115.14 (2001), pp. 6289–6292.

[ACS13] Raman Arora, Andy Cotter, and Nati Srebro. "Stochastic optimization of PCA with capped MSG". In: *Advances in Neural Information Processing Systems*. 2013, pp. 1815–1823.

[Aro+12] Raman Arora et al. "Stochastic optimization for PCA and PLS". In: *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*. IEEE. 2012, pp. 861–868.

[Ati+01] Ali Rana Atilgan et al. "Anisotropy of fluctuation dynamics of proteins with an elastic network model". In: *Biophysical journal* 80.1 (2001), pp. 505–515.

[BAE97] Ivet Bahar, Ali Rana Atilgan, and Burak Erman. "Direct evaluation of thermal fluctuations in proteins using a single-parameter harmonic potential". In: *Folding and Design* 2.3 (1997), pp. 173–181.

[Bah+09] Ivet Bahar et al. "Normal mode analysis of biomolecular structures: functional mechanisms of membrane proteins". In: *Chemical reviews* 110.3 (2009), pp. 1463–1497.

[BB09] Ahmet Bakan and Ivet Bahar. "The intrinsic dynamics of enzymes plays a dominant role in determining the structural changes induced upon inhibitor binding". In: *Proceedings of the National Academy of Sciences* 106.34 (2009), pp. 14349–14354.

[BMB11] Ahmet Bakan, Lidio M Meireles, and Ivet Bahar. "ProDy: protein dynamics inferred from theory and experiments". In: *Bioinformatics* 27.11 (2011), pp. 1575–1577.

[Bal+96] Manel A Balsera et al. "Principal component analysis and long time protein dynamics". In: *The Journal of Physical Chemistry* 100.7 (1996), pp. 2567–2572.

[BDF13] Akshay Balsubramani, Sanjoy Dasgupta, and Yoav Freund. "The fast convergence of incremental PCA". In: *Advances in Neural Information Processing Systems*. 2013, pp. 3174–3182.

[BGH06] Riccardo Baron, Wilfred F van Gunsteren, and Philippe H Hünenberger. "Estimating the configurational entropy from molecular dynamics simulations: anharmonicity and correlation corrections to the quasi-harmonic approximation". In: *Trends Phys Chem* 11 (2006), pp. 87–122.

[BHM09] Riccardo Baron, Philippe H Hünenberger, and J Andrew McCammon. "Absolute single-molecule entropies from quasi-harmonic analysis of microsecond molecular dynamics: correction terms and convergence properties". In: *Journal of chemical theory and computation* 5.12 (2009), pp. 3150–3160.

[BV12] Riccardo Baron and Nadeem A Vellore. “LSD1/CoREST is an allosteric nanoscale clamp regulated by H3-histone-tail molecular recognition”. In: *Proceedings of the National Academy of Sciences* 109.31 (2012), pp. 12509–12514.

[Bas+98a] Gautam Basu et al. “Protein electron transfer reorganization energy spectrum from normal mode analysis. 1. Theory”. In: *The Journal of Physical Chemistry B* 102.11 (1998), pp. 2076–2084.

[Bas+98b] Gautam Basu et al. “Protein electron transfer reorganization energy spectrum from normal mode analysis. 2. Application to Ru-modified cytochrome c”. In: *The Journal of Physical Chemistry B* 102.11 (1998), pp. 2085–2094.

[BMS14] Soumendranath Bhakat, Alberto JM Martin, and Mahmoud ES Soliman. “An integrated molecular dynamics, principal component analysis and residue interaction network approach reveals the impact of M184V mutation on HIV reverse transcriptase resistance to lamivudine”. In: *Molecular BioSystems* 10.8 (2014), pp. 2215–2228.

[BK88] Hervé Bourlard and Yves Kamp. “Auto-association by multilayer perceptrons and singular value decomposition”. In: *Biological cybernetics* 59.4-5 (1988), pp. 291–294.

[Boy+06] Stephen Boyd et al. “Randomized gossip algorithms”. In: *IEEE transactions on information theory* 52.6 (2006), pp. 2508–2530.

[Bri+05] Alessandro Brigo et al. “Comparison of multiple molecular dynamics trajectories calculated for the drug-resistant HIV-1 integrase T66I/M154I catalytic domain”. In: *Biophysical journal* 88.5 (2005), pp. 3072–3082.

[CD18] Hervé Cardot and David Degras. “Online Principal Component Analysis in High Dimension: Which Algorithm to Choose?” In: *International Statistical Review* 86.1 (2018), pp. 29–50.

[Cas+05] David A Case et al. “The Amber biomolecular simulation programs”. In: *Journal of computational chemistry* 26.16 (2005), pp. 1668–1688.

[Cha+15] Tsung-Han Chan et al. “PCANet: A simple deep learning baseline for image classification?” In: *IEEE transactions on image processing* 24.12 (2015), pp. 5017–5032.

[Che+11] Cheng-Kai Chen et al. “Static correlation visualization for large time-varying volume data”. In: *2011 IEEE Pacific Visualization Symposium*. IEEE. 2011, pp. 27–34.

[Cho17] François Chollet. “Xception: Deep learning with depthwise separable convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1251–1258.

[Cla+95] James B Clarage et al. "A sampling problem in molecular dynamics simulations of macromolecules." In: *Proceedings of the National Academy of Sciences* 92.8 (1995), pp. 3288–3292.

[CM07] Adam E Cohen and WE Moerner. "Principal-components analysis of shape fluctuations of single DNA molecules". In: *Proceedings of the National Academy of Sciences* 104.31 (2007), pp. 12622–12627.

[DA12] Isabella Daidone and Andrea Amadei. "Essential dynamics: foundation and applications". In: *Wiley Interdisciplinary Reviews: Computational Molecular Science* 2.5 (2012), pp. 762–770.

[DK93] KI Diamantaras and Sun-Yuan Kung. "Compressing moving pictures using the APEX neural principal component extractor". In: *Neural Networks for Signal Processing III-Proceedings of the 1993 IEEE-SP Workshop*. IEEE. 1993, pp. 321–330.

[DK96] Konstantinos I Diamantaras and Sun Yuan Kung. *Principal component neural networks: theory and applications*. John Wiley & Sons, Inc., 1996.

[Dim+10] Alexandros G Dimakis et al. "Gossip algorithms for distributed signal processing". In: *Proceedings of the IEEE* 98.11 (2010), pp. 1847–1864.

[DS10] Eric C Dykeman and Otto F Sankey. "Normal mode analysis and applications in biological physics". In: *Journal of Physics: Condensed Matter* 22.42 (2010), p. 423202.

[EV13] Ehsan Elhamifar and Rene Vidal. "Sparse subspace clustering: Algorithm, theory, and applications". In: *IEEE transactions on pattern analysis and machine intelligence* 35.11 (2013), pp. 2765–2781.

[EV09] Ehsan Elhamifar and René Vidal. "Sparse subspace clustering". In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. 2009, pp. 2790–2797.

[FPG14] Jérôme Fellus, David Picard, and Philippe-Henri Gosselin. "Dimensionality reduction in decentralized networks by Gossip aggregation of principal components analyzers". In: *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. 2014, pp. 171–176.

[GSS17] Dan Garber, Ohad Shamir, and Nathan Srebro. "Communication-efficient Algorithms for Distributed Stochastic Principal Component Analysis". In: *arXiv preprint arXiv:1702.08169* (2017).

[GK04] Jeff Gill and Gary King. "What to do when your Hessian is not invertible: Alternatives to model respecification in nonlinear estimation". In: *Sociological methods & research* 33.1 (2004), pp. 54–87.

[Gio+17] Valentina Giorgio et al. "Ca2+ binding to F-ATP synthase β subunit triggers the mitochondrial permeability transition". In: *EMBO reports* 18.7 (2017), pp. 1065–1076.

[GB10] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010, pp. 249–256.

[GW69] R Gnanadesikan and MB Wilk. "Data analytic methods in multivariate statistical analysis". In: *Multivariate analysis* 2 (1969), pp. 593–638.

[GK72] Ramanathan Gnanadesikan and John R Kettenring. "Robust estimates, residuals, and outlier detection with multiresponse data". In: *Biometrics* (1972), pp. 81–124.

[GV12] Gene H Golub and Charles F Van Loan. *Matrix computations*. Vol. 3. JHU Press, 2012.

[Gra83] Jørgen Pedersen Gram. "Ueber die Entwickelung reeller Functionen in Reihen mittelst der Methode der kleinsten Quadrate." In: *Journal für die reine und angewandte Mathematik* 94 (1883), pp. 41–73.

[Gue+18] Lionel Gueguen et al. "Faster neural networks straight from JPEG". In: *Advances in Neural Information Processing Systems*. 2018, pp. 3933–3944.

[HL13] Fang Han and Han Liu. "Principal component analysis on non-Gaussian dependent data". In: *International Conference on Machine Learning*. 2013, pp. 240–248.

[HP14] Moritz Hardt and Eric Price. "The noisy power method: A meta algorithm with applications". In: *Advances in Neural Information Processing Systems*. 2014, pp. 2861–2869.

[HKG94] Steven Hayward, Akio Kitao, and Nobuhiro Gō. "Harmonic and anharmonic aspects in the dynamics of BPTI: a normal mode analysis and principal component analysis". In: *Protein Science* 3.6 (1994), pp. 936–943.

[He+16] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[Heb49] Donald Olding Hebb. *The organization of behavior*. New York: Wiley Sons, 1949.

[Her45] Gerhard Herzberg. *Infrared and Raman spectra of polyatomic molecules*. D. Van Nostrand Company.; New York, 1945.

[HS06] Geoffrey E Hinton and Ruslan R Salakhutdinov. "Reducing the dimensionality of data with neural networks". In: *science* 313.5786 (2006), pp. 504–507.

[Hon11] Paul Honeine. "Online kernel principal component analysis: A reduced-order model". In: *IEEE transactions on pattern analysis and machine intelligence* 34.9 (2011), pp. 1814–1826.

[Hua+17] Gao Huang et al. "Densely connected convolutional networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708.

[HJL07] Gary B. Huang, Vidit Jain, and Erik Learned-Miller. "Unsupervised Joint Alignment of Complex Images". In: *ICCV*. 2007.

[Hua+07] Gary B. Huang et al. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. Tech. rep. 07-49. University of Massachusetts, Amherst, Oct. 2007.

[Hua+12] Gary Huang et al. "Learning to align from scratch". In: *Advances in neural information processing systems*. 2012, pp. 764–772.

[Ing+14] Helgi I Ingólfsson et al. "The power of coarse graining in biomolecular simulations". In: *Wiley Interdisciplinary Reviews: Computational Molecular Science* 4.3 (2014), pp. 225–248.

[IS15] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167* (2015).

[IM09] Anthony Ivetac and J Andrew McCammon. "Elucidating the inhibition mechanism of HIV-1 non-nucleoside reverse transcriptase inhibitors through multicopy molecular dynamics simulations". In: *Journal of molecular biology* 388.3 (2009), pp. 644–658.

[IS09] Alan J Izenman and Yan Shen. "Outlier detection using the smallest kernel principal components". In: *Astro Temple Edu, pdf report* (2009).

[JEG12] Yun Jang, David S Ebert, and Kelly Gaither. "Time-varying data visualization using functional representations". In: *IEEE Transactions on Visualization and Computer Graphics* 18.3 (2012), pp. 421–433.

[JZ13] Rie Johnson and Tong Zhang. "Accelerating stochastic gradient descent using predictive variance reduction". In: *Advances in neural information processing systems*. 2013, pp. 315–323.

[Jol02] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

[JR05] Alark Joshi and Penny Rheingans. "Illustration-inspired techniques for visualizing time-varying data". In: *VIS 05. IEEE Visualization, 2005.* IEEE. 2005, pp. 679–686.

[Kag] Kaggle. *Sign Language MNIST*. https://www.kaggle.com/datamunge/sign-language-mnist. Available from Kaggle dataset repository.

[KW10] Shina CL Kamerlin and Arieh Warshel. "At the dawn of the 21st century: Is dynamics the missing link for understanding enzyme catalysis?" In: *Proteins: Structure, Function, and Bioinformatics* 78.6 (2010), pp. 1339–1375.

[KVW14] Ravi Kannan, Santosh Vempala, and David Woodruff. "Principal component analysis and higher correlations for distributed data". In: *Conference on Learning Theory*. 2014, pp. 1040–1057.

[Key81] Robert Keys. "Cubic convolution interpolation for digital image processing". In: *IEEE transactions on acoustics, speech, and signal processing* 29.6 (1981), pp. 1153–1160.

[Kmi+16] Sebastian Kmiecik et al. "Coarse-grained protein models and their applications". In: *Chemical Reviews* 116.14 (2016), pp. 7898–7936.

[Kor08] Jesse D Kornblum. "Using JPEG quantization tables to identify imagery processed by software". In: *digital Investigation* 5 (2008), S21–S25.

[Kra69] TP Krasulina. "A method of stochastic approximation for the determination of the least eigenvalue of a symmetric matrix". In: *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki* 9.6 (1969), pp. 1383–1387.

[KH+09] Alex Krizhevsky, Geoffrey Hinton, et al. *Learning multiple layers of features from tiny images*. Tech. rep. University of Toronto, 2009.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[Lea14] Gary B. Huang Erik Learned-Miller. *Labeled Faces in the Wild: Updates and New Reporting Procedures*. Tech. rep. UM-CS-2014-003. University of Massachusetts, Amherst, May 2014.

[LeC98] Yann LeCun. "The MNIST database of handwritten digits". In: *http://yann. lecun. com/exdb/mnist/* (1998).

[LLL16] Chun-Liang Li, Hsuan-Tien Lin, and Chi-Jen Lu. "Rivalry of two families of algorithms for memory-restricted streaming pca". In: *Artificial Intelligence and Statistics*. 2016, pp. 473–481.

[Liu+14] Shusen Liu et al. "Multivariate volume visualization through dynamic projections". In: *2014 IEEE 4th Symposium on Large Data Analysis and Visualization (LDAV)*. IEEE. 2014, pp. 35–42.

[Liu+15] Ziwei Liu et al. "Deep Learning Face Attributes in the Wild". In: *Proceedings of International Conference on Computer Vision (ICCV)*. 2015.

[LS92] Siegrid Lowel and Wolf Singer. "Selection of intrinsic horizontal connections in the visual cortex by correlated neuronal activity". In: *Science* 255.5041 (1992), pp. 209–212.

[MLS09] Gia G Maisuradze, Adam Liwo, and Harold A Scheraga. "Principal component analysis for protein folding dynamics". In: *Journal of molecular biology* 385.1 (2009), pp. 312–329.

[McG+15] Robert T McGibbon et al. "MDTraj: a modern open library for the analysis of molecular dynamics trajectories". In: *Biophysical journal* 109.8 (2015), pp. 1528–1532.

[MP29] RV Mises and Hilda Pollaczek-Geiringer. "Praktische Verfahren der Gleichungsauflösung." In: *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik* 9.2 (1929), pp. 152–164.

[MCJ13] Ioannis Mitliagkas, Constantine Caramanis, and Prateek Jain. "Memory limited, streaming PCA". In: *Advances in Neural Information Processing Systems*. 2013, pp. 2886–2894.

[Mon04] John Mongan. "Interactive essential dynamics". In: *Journal of computer-aided molecular design* 18.6 (2004), pp. 433–436.

[MR05] Diana Mustard and David W Ritchie. "Docking essential dynamics eigenstructures". In: *Proteins: Structure, function, and bioinformatics* 60.2 (2005), pp. 269–274.

[NSI99] Ko Nishino, Yoichi Sato, and Katsushi Ikeuchi. "Eigen-texture method: Appearance compression based on 3D model". In: *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.* Vol. 1. IEEE. 1999.

[Nvi] CUDA Nvidia. "Programming Guide, NVIDIA Corporation 2012". In: ().

[Oja82] Erkki Oja. "Simplified neuron model as a principal component analyzer". In: *Journal of mathematical biology* 15.3 (1982), pp. 267–273.

[OJE83] E OJE. "Subspace methods of pattern recognition". In: *Pattern recognition and image processing series*. Vol. 6. Research Studies Press. 1983.

[Oro+03] Modesto Orozco et al. "Theoretical methods for the simulation of nucleic acids". In: *Chemical Society Reviews* 32.6 (2003), pp. 350–364.

[Pea01] Karl Pearson. "LIII. On lines and planes of closest fit to systems of points in space". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), pp. 559–572.

[PSC18] Cengiz Pehlevan, Anirvan M Sengupta, and Dmitri B Chklovskii. "Why do similarity matching objectives lead to Hebbian/anti-Hebbian networks?" In: *Neural computation* 30.1 (2018), pp. 84–124.

[Plu95] Mark D Plumbley. “Lyapunov functions for convergence of principal component algorithms”. In: *Neural Networks* 8.1 (1995), pp. 11–23.

[Qu+02] Yongming Qu et al. “Principal component analysis for dimension reduction in massive distributed data sets”. In: *Proceedings of IEEE International Conference on Data Mining (ICDM)*. 2002.

[RA09] Arvind Ramanathan and Pratul K Agarwal. “Computational identification of slow conformational fluctuations in proteins”. In: *The Journal of Physical Chemistry B* 113.52 (2009), pp. 16669–16680.

[Rao+97] AM Rao et al. “Diameter-selective Raman scattering from vibrational modes in carbon nanotubes”. In: *Science* 275.5297 (1997), pp. 187–191.

[RM85] Herbert Robbins and Sutton Monro. “A stochastic approximation method”. In: *Herbert Robbins Selected Papers*. Springer, 1985, pp. 102–109.

[Rus+15] Olga Russakovsky et al. “Imagenet large scale visual recognition challenge”. In: *International journal of computer vision* 115.3 (2015), pp. 211–252.

[SMG00] Heiko Schäfer, Alan E Mark, and Wilfred F van Gunsteren. “Absolute entropies from molecular dynamics simulation trajectories”. In: *The Journal of Chemical Physics* 113.18 (2000), pp. 7809–7817.

[Sch07] E Schmidt. “Zür Theorie der linearen und nichtlinearen Integralgleichungen. I Teil. Entwicklung willkurlichen Funktionen nach System vorgeschriebener”. In: *Math. Ann* 63 (1907), pp. 161–174.

[SSM98] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. “Nonlinear component analysis as a kernel eigenvalue problem”. In: *Neural computation* 10.5 (1998), pp. 1299–1319.

[Sha15] Ohad Shamir. “A Stochastic PCA and SVD Algorithm with an Exponential Convergence Rate.” In: *ICML*. 2015, pp. 144–152.

[SXZ12] Prathamesh M Shenai, Zhiping Xu, and Yang Zhao. “Applications of principal component analysis (PCA) in materials science”. In: *Principal component analysis-engineering applications*. IntechOpen, 2012.

[Sil] D. Silver. *Turbulent Vortex Dataset*. http://www.cs.ucdavis.edu/~ma/ITR. Available from the Time-varying data repository at UCDavis.

[SZ14] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[Skj+14] Lars Skjærven et al. “Integrating protein structural dynamics and evolutionary analysis with Bio3D”. In: *BMC bioinformatics* 15.1 (2014), p. 399.

[Smi17] Leslie N Smith. "Cyclical learning rates for training neural networks". In: *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2017, pp. 464–472.

[Sze+15] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[Sze+16] Christian Szegedy et al. "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[Sze+17] Christian Szegedy et al. "Inception-v4, inception-resnet and the impact of residual connections on learning". In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

[TB04] Ian F Thorpe and Charles L Brooks III. "The coupling of structural fluctuations to hydride transfer in dihydrofolate reductase". In: *Proteins: Structure, Function, and Bioinformatics* 57.3 (2004), pp. 444–457.

[Tir96] Monique M Tirion. "Large amplitude elastic motions in proteins from a single-parameter, atomic analysis". In: *Physical Review Letters* 77.9 (1996), p. 1905.

[Tre11] Nick Trefethen. "Favorite eigenvalue problems". In: *SIAM News* 44.10 (2011).

[Tsi84] John Nikolas Tsitsiklis. *Problems in decentralized decision making and computation.* Tech. rep. MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR INFORMATION and DECISION SYSTEMS, 1984.

[Vid11] René Vidal. "Subspace clustering". In: *IEEE Signal Processing Magazine* 28.2 (2011), pp. 52–68.

[VF14] René Vidal and Paolo Favaro. "Low rank subspace clustering (LRSC)". In: *Pattern Recognition Letters* 43 (2014), pp. 47–61.

[VMP04] René Vidal, Yi Ma, and Jacopo Piazzi. "A new GPCA algorithm for clustering subspaces by fitting, differentiating and dividing polynomials". In: *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*. Vol. 1. IEEE. 2004, pp. I–I.

[Vin09] Valda J Vinson. "Proteins in motion". In: *Science* 324.5924 (2009), pp. 197–197.

[Vos+91] Marten H Vos et al. "Direct observation of vibrational coherence in bacterial reaction centers using femtosecond absorption spectroscopy." In: *Proceedings of the National Academy of Sciences* 88.20 (1991), pp. 8885–8889.

[Wal92] Gregory K Wallace. "The JPEG still picture compression standard". In: *IEEE transactions on consumer electronics* 38.1 (1992), pp. xviii–xxxiv.

[WYM08] Chaoli Wang, Hongfeng Yu, and Kwan-Liu Ma. "Importance-driven time-varying data visualization". In: *IEEE Transactions on Visualization and Computer Graphics* 14.6 (2008), pp. 1547–1554.

[WZH03] Juyang Weng, Yilu Zhang, and Wey-Shiuan Hwang. "Candid covariance-free incremental principal component analysis". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25.8 (2003), pp. 1034–1040.

[Wer74] P Werbos. "New tools for prediction and analysis in the behavioral science [dissertation]". In: *Harvard University Press* (1974).

[Wu+18] Sissi Xiaoxiao Wu et al. "A Review of Distributed Algorithms for Principal Component Analysis". In: *Proceedings of the IEEE* 106.8 (2018), pp. 1321–1340.

[XRV17] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. Aug. 28, 2017. arXiv: `cs.LG/1708.07747 [cs.LG]`.

[Xie+17] Saining Xie et al. "Aggregated residual transformations for deep neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1492–1500.

[Xu+18] Peng Xu et al. "Accelerated Stochastic Power Iteration". In: *International Conference on Artificial Intelligence and Statistics*. 2018, pp. 58–67.

[Yan+09] Lee-Wei Yang et al. "Principal component analysis of native ensembles of biomolecular structures (PCA_NEST): insights into functional dynamics". In: *Bioinformatics* 25.5 (2009), pp. 606–614.

[YSJ09] Lei Yang, Guang Song, and Robert L Jernigan. "Protein elastic network models and the ranges of cooperativity". In: *Proceedings of the National Academy of Sciences* 106.30 (2009), pp. 12347–12352.

[YJL04] Jieping Ye, Ravi Janardan, and Qi Li. "GPCA: an efficient dimension reduction scheme for image compression and retrieval". In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2004, pp. 354–363.

[Yu+15] Fisher Yu et al. "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop". In: *arXiv preprint arXiv:1506.03365* (2015).

[ZK16] Sergey Zagoruyko and Nikos Komodakis. "Wide residual networks". In: *arXiv preprint arXiv:1605.07146* (2016).

[ZG06] Mu Zhu and Ali Ghodsi. "Automatic dimensionality selection from the scree plot via the use of profile likelihood". In: *Computational Statistics & Data Analysis* 51.2 (2006), pp. 918–930.