Imperial College London

Department of Electrical and Electronic Engineering

Final Year Project Report 2021

Project Title:	**Tensor Methods for Portfolio Construction Using Factor Methods**
Student:	**Snehil Kumar**
CID:	**01355676**
Course:	**4EM**
Project Supervisor:	**Prof. Danilo Mandic**
Second Marker:	**Dr. Cong Ling**

Abstract

This project investigates the application of Tensor Decomposition methods such as Canonical Polyadic Decomposition (CPD) and Tucker Decomposition including Higher-Order Singular Value Decomposition (HOSVD) and Higher-Order Orthogonal Iteration (HOOI) algorithms for combining stocks market data and financial fundamentals data. The state of art methods use statistical analysis techniques including Principal Component Analysis (PCA) and Independent Component Analysis (ICA) to select stocks for portfolio construction. The tensor-based approach discussed in the project tensorizes the stocks data and combines tensor decomposition methods and statistical methods together for weighing or selecting stocks for portfolio construction. The thesis begins with a basic background behind statistical analysis and tensor decomposition methods and discusses their practical application and implementation. The statistical and tensor-based approaches are applied on financial data for 20 technology stocks to construct portfolios and evaluate their performance in terms of various trading performance metrics. Overall, tensor decomposition methods can be used to construct portfolios when combined with statistical analysis for finding dominant assets. The experimental results for the tensor-based approach and conventional statistical techniques are compared and the possibilities for future research are discussed.

Acknowledgements

I would like to express my deep and sincere gratitude to my principal supervisor, Professor Danilo Mandic, Researcher and Senior Lecturer with the Department of Electrical and Electronic Engineering, Imperial College London whose experience for providing invaluable support, guidance and feedback throughout the research.

I would also like to thank my second marker, Dr. Cong Ling, Senior Lecturer in the Electrical and Electronic Engineering Department at Imperial College London, and Prof. Tony Constantinides, Emeritus Professor in the Electrical and Electronic Engineering Department at Imperial College London, whose ideas and opinions helped me to complete my project and prepare my thesis.

I am also grateful to Yao Lei Xu, PhD student at Imperial College London, and Mahmoud Mahfouz, Masters student at Imperial College London, for offering their invaluable help, discussion and advice for the project.

In addition, I would also like to thank my family and friends for their unconditional love and support, and the Faculty of Engineering at Imperial College London for inspiring me over the past 4 years.

Contents

CHAPTER 1

Introduction

Multidimensional data is becoming ubiquitous across different industries- from science and engineering to healthcare and finance. Analysing multidimensional data allows us to assess information from different points of view by following a clear and objective approach. The advantage of multidimensional data analysis is observing trends in data across various entities and identifying key factors that can generate good results. As the world enters the era of Machine Learning, Signal Processing and Data Science, many applications require huge amount of structured high-dimensional input data. Big Data requires processing of high volume, variety and complexity of information. Existing standard methods and algorithms become inadequate for the processing and optimisation of such data. As the size of dataset increases, the complexity of Machine Learning and Signal Processing algorithms increases exponentially. Moreover, many problems nowadays require techniques that work not only in large volume, but can solve other challenges of big datasets, i.e., robustness against noisy and inconsistent data, integration of a variety of data types, processing streams of data close to real-time and extracting high quality and consistent data for meaningful and interpretable results. Using tensors instead of the original input vectors, as in the case of conventional two-dimensional analysis including Principal Component Analysis (PCA) and Independent Component Analysis (ICA), can effectively solve these high-dimensional problems. [1] [2]

Tensors are multi-dimensional generalisation of matrices that often provide a natural, sparse and distributed representation for complex data. Tensors have been adopted in diverse branches of data analysis, such as Signal and Image processing, Psychometric, Chemometrics, Biometric, Quantum Physics/Information, Quantum Chemistry and Brain Science [3][4][5][6][7][8][9][10]. Tensors are also suitable for problems in bio- and neuroinformatics or computational neuroscience where data is collected in various forms of big, sparse graphs or networks with multiple aspects and high dimensionality.

In terms of application, the project will focus on financial data which can be represented as high-dimensional tensors. Financial datasets, such as daily stock prices, volume and equity fundamentals are large and complex which makes them susceptible to the curse of dimensionality. Though there are certain reshaping and mathematical techniques that allow us to perform analysis on these datasets in two-dimensional (2D) space, some inherent characteristics can be inevitably lost in the process. This project will explore the application of Tensor decomposition methods to solve these challenges and increase interpretability of financial data.

Section 2 provides an overview on the application of PCA for analysing big data and explores related literature on tensors, which includes tensor operations, multilinear algebra, notations, decomposition methods, algorithms and application. Financial concepts and terminologies relevant to the project are also explained. Section 3 includes the list of selected stocks for analysis and processes the closing prices into log returns before applying different statistical and tensor-based techniques for constructing the portfolio. Section 4 applies two-dimensional statistical analysis methods such PCA and ICA on the returns data for constructing the portfolio while section 5 discusses the application of a tensor-based approach which integrates statistical analysis for selecting or weighing stocks. Section

6 evaluates the results obtained from Section 4 and 5 to identify the best performing portfolios. Finally, section 7 concludes our findings and discusses the potential for further research and application that can help gain more intuition on the viability of tensor decomposition methods for financial analysis and improve the performance of portfolios. The success of the project depends on the ability to identify the best methods for portfolio construction that can help improve the performance of financial models on multidimensional data.

CHAPTER 2

Background

This chapter introduces the fundamental concepts behind statistical analysis techniques, tensor decomposition methods and financial terminologies pertinent to the project. Moreover, it further provides detailed background mathematical framework and measures leading to an introduction on tensors and the financial implementation related to the aims of the project.

2.1 Principal Component Analysis

Principal Component Analysis, also called PCA, is a mathematical technique commonly used to reduce the complexity of a dataset and minimise information loss to improve analysis of high-dimensional data. It transforms a dataset with a large number of interrelated variables into a new combination of uncorrelated variables, called the principal components, arranged in the decreasing order of variance. Each principal component is a linear combination of the original variables in which the coefficients indicate the relative importance of the variable in the component. The first principal component has the maximal variance and plays a predominant role in analysing movements in the dataset. A detailed explanation on the application of PCA in Data Analysis can be found in [11].

When the data are tensor objects, PCA vectorises the objects into a long vector and generates a design matrix. Conventional PCA may lead to inefficient and unstable prediction due to its extremely large dimensionality. For example, if there are 400 images in a dataset and each contains 64 x 64 pixels, vectorising the images would contribute to a design matrix of size 400 x 4096. To overcome these limitations, a Multilinear PCA, or MPCA was proposed that tries to preserve the data structure by searching for low-dimensional multilinear projections, hence decreasing the dimensionality efficiently. The Tucker Decomposition method for Tensors is a direct generalisation of Multilinear Principal Component Analysis which will be explained in detail in section 3.2.3. A deeper assessment on Conventional and Multilinear PCA is provided in [12]. For portfolio management, PCA is accompanied by two rules: Cumulative Variance and Kaiser's Rule, to find the number of principal components to retain. The application of PCA to stock portfolio management and the analysis of results using Cumulative Variance and Kaiser's Rule are thoroughly covered in [13][14].

Nowadays, many researchers are exploring techniques to reduce the complexity of high-dimensional data and deduce relevant and accurate information from statistical analysis. Tensors, which have a huge potential to solve the limitations of PCA, have been discussed in section 2.3.

2.2 Independent Component Analysis

Independent Component Analysis, also called ICA, is a signal processing technique that takes a set of signal vectors and extracts a set of statistically independent vectors, knows as Independent Components (ICs) or the sources. They are estimated as the original source signals which are assumed to be mixed linearly to form the observed signal, such that it can be expressed as $x = As$. Figure 2.2.1 shows a schematic representation of ICA. The original sources s are mixed through matrix A to form the observed signal x. The demixing matrix W transforms the observed signal x into the independent components y. [15]

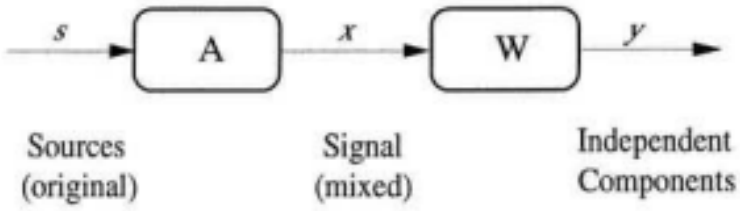

Figure 2.2.1: Schematic representation of ICA. Reprinted from "A first application of independent component analysis to extracting structure from stock returns" by Andrew D. Back and Andreas S. Weigend, 1997, International Journal of Neural Systems, p.475.

The schematic representation can be expressed as $x_i(t) = \sum_{j=1}^{n} a_{ij} s_j(t)$, where $x = [x_1(t), ..., x_n(t)]$ is the mixtures, $s = [s_1(t), ..., s_n(t)]$ is the independent sources and A is the n × n unknown mixing matrix of real number. ICA aims to find the unmixing matrix W such that $s = Wx$ becomes as independent as possible. The ability to estimate W depends on the following conditions: [16][17]

- The sources must be statistically independent.
- The sources must have non-Gaussian distributions.
- The number of mixtures N must be at least the same as the number of the independent components M.
- The mixtures must be a linear combination of the independent sources.
- The signals are stationary.

Although ICA and PCA seem to be related, they perform different tasks. PCA aims to compress information through dimensionality reduction, while ICA aims to separate information by transforming the input space into a maximally independent basis. The principal components are uncorrelated and gives projections of the data in the direction of the maximum variance. The principal components (PCs) are ordered in terms of their variances. ICA seeks to obtain statistically independent components where the order, amplitude and sign of the ICs are ambiguous. Moreover, PCA algorithms use only second order statistical information, while ICA algorithms may use higher order statistical information for separating the signals. [15][18]

For portfolio construction, ICA can find interpretable factors of instantaneous stock returns which can be useful to select dominant stocks.

2.3 Tensors

2.3.1 Tensor Basics, Notations and Operations

A tensor is a multilinear generalisation of a matrix or a vector, i.e., it could be a 0-D matrix (a single number), a 1-D matrix (a vector), a 3-D matrix (something like a cube of numbers), or a higher dimensional structure that is harder to visualise.

Mathematically, a d^{th}-order tensor is simply a d-dimensional array of complex valued data $\chi \in R^{n_1 \times n_2 \times .... \times n_d}$ for given dimension sizes $n_1, n_2, ..., n_d \in N$, where each of the dimension is conventionally referred to as mode or rank. For each of the k^{th} mode where k = 1, 2, . . . , d, it has a total of p_k elements that can be indexed using l_k = 1,2, . . . ,p_k. Therefore, this data structure can be simply regarded as a multi-dimensional array where each element of the tensor is defined with d indices $\chi[l_1, ..., l_d]$.

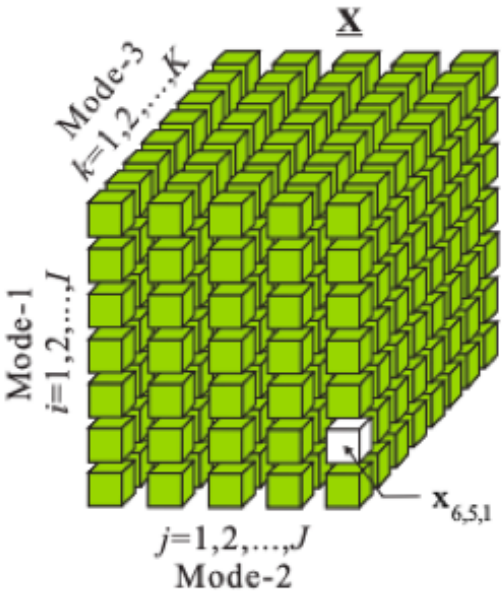

Figure 2.3.1: A 3^{rd}-order tensor $\underline{X} \in R^{I \times J \times K}$ with entries $x_{ijk} = \underline{X}$(i, j, k) and exemplary symbols used in tensor network diagrams. Each node in the diagram represents a tensor and each edge represents a mode or dimension. We indicate maximum size in each mode by I, J, K or running indices: i = 1, 2, . . . , I; j = 1, 2, . . . , J and k = 1, 2, . . . , K. Reprinted from "Tensor Networks for Big Data Analytics and Large-Scale Optimization Problems" by Andrzej Cichocki, 2014, CoRR, abs/1407.3124, p.3. Copyright by ArXiv.

Tensor Fiber and Slice

A tensor fiber is a one-dimensional fragment of a tensor, obtained by fixing all indices except one. Tensor fibers are higher-order extension of matrix rows and columns. A third-order tensor has fibers that can be denoted by $x_{:jk}$, $x_{i:k}$ or $x_{ij:}$ correspondingly.

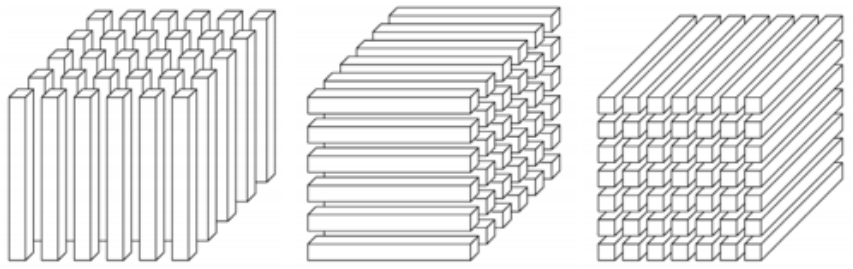

Figure 2.3.2: Fibers of a 3^{rd} order tensor. Reprinted from "Tensor decompositions and applications" by T. Kolda and B. Bader, 2009, SIAM Review, 51(3), p.458. Copyright by SIAM.

A tensor slice is a two-dimensional fragment of a tensor, obtained by fixing all indices except two. For instance, $X_{i::}$, $X_{:j:}$ and $X_{::k}$ denote the horizontal, lateral, and frontal slices of a 3-way tensor, respectively. [19]

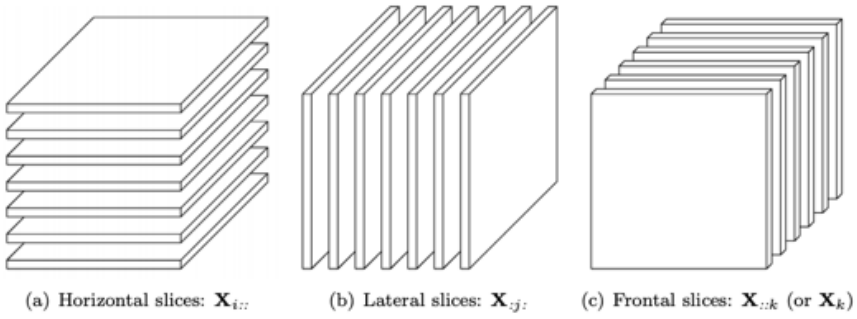

Figure 2.3.3: Slices of a 3^{rd} order tensor. Reprinted from "Tensor decompositions and applications" by T. Kolda and B. Bader, 2009, SIAM Review, 51(3), p.458. Copyright by SIAM.

$x = [X_N], X = [X_{N_1,N_2}], \chi = [X_{N_1,...,N_P}]$	Vector, Matrix and Tensor
x^*	Complex conjugate
$\Lambda \in R^{R \times R \times ... \times R}$	Diagonal core tensor with nonzero entries λ_r on main diagonal
$X^1, X^T, X^\dagger$	Inverse, Transpose and Moore-Penrose pseudo inverse
$\circ$	Outer product
$\bigtriangledown$	Kronecker product
$*$	Hadamard product
$\odot$	Khatri-Rao product
x_N	Mode-n product
$X_{(N)} \in R^{I_N I_1 ... I_{N1} I_{N+1} ... I_N}$	Mode-n matricization of tensor χ
$[[A, B, C]]$	Canonical Polyadic (Parafac) decomposition
$\chi(:, I_2, I_3, ..., I_N)$	Mode-1 fiber of a tensor
$\chi(:, :, I_3, ..., I_N)$	Slice of a tensor

Table 2.3.1: Tensor and Multi-linear algebra notation used in this project

Rank-One Tensors
An N-way tensor $\chi \in R^{I_1 \times I_2 \times ... \times I_N}$ is rank one if it can be written as the outer product of N vectors, i.e.,

$$\chi = x(1) \circ x(2) \circ ... \circ x(N) \tag{2.3.1}$$

The symbol "$\circ$" represents the vector outer product, as stated in Table 2.3.1. This means that each element of the tensor is the product of the corresponding vector elements:

$$x_{i_1 i_2 ... i_N} = a^{(1)}_{i_1} a^{(2)}_{i_2} ... a^{(N)}_{i_N}, \tag{2.3.2}$$

for all $1 \leqslant i_n \leqslant I_n$.

Supersymmetric Tensors
A tensor is called cubical if every mode is the same size, i.e., $\chi \in R^{I \times I \times I \times ... \times I}$. A cubical tensor is called supersymmetric if its elements remain constant under any permutation of the indices. For instance, a three-way tensor $\chi \in R^{I \times I \times I}$ is supersymmetric if,

$$x_{ijk} = x_{ikj} = x_{jik} = x_{jki} = x_{kij} = x_{kji}, \tag{2.3.3}$$

for all i, j, k = 1,...,I.

Frobenius Norm

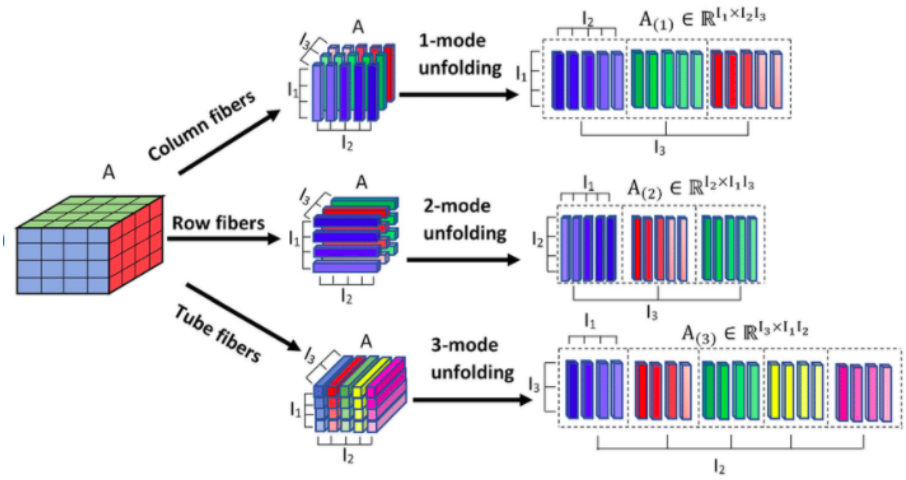

Figure 2.3.4: Example of unfolding tensor A along 1-mode, 2-mode and 3-mode. Reprinted from "Multimodal 2d + 3d multi-descriptor tensor for face verification" by Adel Saoud, Abdelmalik Ouamane, Abdelkrim Ouafi and Abdelmalik Taleb-Ahmed, 2020, Multimedia Tools and Applications, p.5. Copyright by Springer Science+Business Media, LLC, part of Springer Nature 2020.

The Frobenius Norm of a Tensor $\chi \in R^{I_1 \times I_2 \ldots \times I_N}$ is the square root of the sum of the squares of all its elements:

$$||\chi|| = \sqrt{\sum_{i_1}^{I_1} \sum_{i_2}^{I_2} \ldots \sum_{i_N}^{I_N} \chi(i_1, i_2, \ldots i_n)^2} \tag{2.3.4}$$

Inner Product
The inner product of two same-sized tensors $\chi, \Upsilon \in R^{I_1 \times I_2 \times \ldots \times I_N}$ is the sum of the products of their entries,

$$|| < \chi, \Upsilon || = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \ldots \sum_{i_N=1}^{I_N} x_{i_1, i_2, \ldots i_N} y_{i_1, i_2, \ldots i_N} \tag{2.3.5}$$

Moreover, one can say that $< \chi, \chi >= ||\chi||^2$.

Matricization
N-mode Matricization is simply the unfolding of a tensor to a matrix in N different ways. For instance, a 2×3×4 tensor can be arranged as a 6×4 matrix or a 3 × 8 matrix, and so on. Tensor element $(i_1, i_2, ..., i_N)$ maps to matrix element (i_n, j), where,

$$j = 1 + \sum_{\substack{k=1 \\ k \neq n}}^{N} (i_k - 1)\, J_k \tag{2.3.6}$$

with $J_k = \prod_{\substack{m=1 \\ m \neq n}}^{k-1} I_m$

N-mode Tensors and their Products
The n-mode (matrix) product of a tensor χ with a matrix U is denoted by $\chi\ x_n$ U,

$$(\boldsymbol{X} \times_n \mathbf{U})_{i_1 \ldots i_{n-1} j i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \cdots i_N} u_{j i_n} \tag{2.3.7}$$

The n-mode (vector) product of a tensor χ with a vector v is denoted by $\chi\ x_n^-$ v,

$$(\boldsymbol{X} \bar{x}_n \mathbf{v})_{i_1 \cdots i_{n-1} i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \cdots i_N} v_{i_n} \tag{2.3.8}$$

Kronecker Product
The Kronecker product of two matrices A $\in R^{I\times J}$ and B $\in R^{K\times L}$ is a matrix C $\in R^{IK\times JL}$,

$$C = A \otimes B = \begin{bmatrix} a_{11}\boldsymbol{B} & a_{12}\boldsymbol{B} & \dots & a_{1J}\boldsymbol{B} \\ a_{21}\boldsymbol{B} & a_{22}\boldsymbol{B} & \dots & a_{2J}\boldsymbol{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}\boldsymbol{B} & a_{I2}\boldsymbol{B} & \dots & a_{IJ}\boldsymbol{B} \end{bmatrix} \tag{2.3.9}$$

Hadamard Product
The Hadamard product is obtained by multiplying the matrices A $\in R^{I\times J}$ and B $\in R^{I\times J}$ element-wise to give matrix C $\in R^{I\times J}$,

$$C = A * B = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & \dots & a_{1J}b_{1J} \\ a_{21}b_{21} & a_{22}b_{22} & \dots & a_{2J}b_{2J} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}b_{I1} & a_{I2}b_{I2} & \dots & a_{IJ}b_{I2} \end{bmatrix} \tag{2.3.10}$$

Khatri-Rao Product
The Khatri-Rao product is similar to the Kronecker product, but it performs the product on the same columns. Therefore, for A $\in^{I\times K}$ and B $\in R^{J\times K}$ to give matrix C $\in R^{IJ\times K}$,

$$C = A \otimes B = \begin{bmatrix} a_{11}b_1 & a_{12}b_2 & \dots & a_{1K}b_K \\ a_{21}b_1 & a_{22}b_2 & \dots & a_{2K}b_K \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}b_1 & a_{I2}b_2 & \dots & a_{IK}b_K \end{bmatrix} \tag{2.3.11}$$

$$\mathrm{A} \circ \mathrm{B} = [\mathrm{a}_1 \otimes \mathrm{b}_1 \mathrm{a}2 \otimes \mathrm{b}_2 \dots \mathrm{a}_\mathrm{K} \otimes \mathrm{b}_\mathrm{K}] \tag{2.3.12}$$

2.3.1.1 Tensorisation

Tensorisation refers to the generation of higher-order structured tensors from the lower-order data formats (e.g., vectors, matrices or low-order tensors). A review on tensorisation techniques can be found in [20]. Tensorised data can be divided into four categories [21]:

- **Rearrangement of lower dimensional data structures:** Large-scale vectors or matrices are readily tensorised to higher-order tensors, and can be compressed through tensor decomposition if they admit a low-rank tensor approximation. This principle facilitates big data analysis.
- **Mathematical construction:** Tensors are acquired using mathematically transformation. For instance, a second order tensor obtained from slices of a third-order tensor. Secondly, a (channel × time) data matrix can be transformed into a (channel × time × frequency) or (channel × time × scale) tensor via time-frequency or wavelet representations.
- **Experiment design:** Multi-faceted data can be naturally stacked into a tensor. For instance, the common modes in EEG recordings across subjects, trials, and conditions are best analyzed when combined together into a tensor.
- **Naturally tensor data:** Some data sources are readily generated as tensors such as Financial data and RGB colour images data.

Reshaping or Folding
This is the simplest way of tensorisation and is also known as segmentation. It is obtained through the folding of a vector by rearranging and reshaping data entries [22]. This type of tensorisation preserves the number of original data entries and their sequential ordering, as it only rearranges a vector to a matrix or tensor. Hence, folding does not require additional memory space. In folding, a tensor χ is obtained from a vector x such that:

$$\begin{gathered} \mathrm{X}\left(\mathrm{i}_1, \mathrm{i}_2, \ldots, \mathrm{i}_\mathrm{N}\right) = \mathbf{x}(\mathrm{i}) \\ \forall 1 \leq \mathrm{i}_\mathrm{n} \leq \mathrm{I}_\mathrm{n} \end{gathered} \tag{2.3.13}$$

where $i = 1 + \sum_{n=1}^{N} (i_n - 1) \prod_{k=1}^{n-1} I_k$ is a linear index of $(i_1, i_2, ...)$

Toeplitz Folding
A Toeplitz matrix has the same entries in each diagonal, such that it obtains the following structure from a vector a of length $L = I + J - 1$:

$$A_{I,J}(y) = \begin{bmatrix} a_I & a_{I+1} & \ldots & a_L \\ a_{I-1} & a_I & \ldots & a_{L-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_1 & a_2 & \ldots & a_{L-I+1} \end{bmatrix} \tag{2.3.14}$$

The first column and first row of the Toeplitz matrix represent its entire generating vector. The Toeplitz tensor can be derived by considering the discrete convolution between vectors, as described in [23]. It is defined as follows:

$$\chi\left(i_1, i_2, \ldots, i_N\right) = a\left(i_1^- + i_2^- + \ldots + i_{N-1}^- + i_N\right), \tag{2.3.15}$$

where $i_n^- = I_n - i_n$

Hankel Folding
An I ×J Hankel matrix of length $I + J1$ is as follows:

$$A_{I,J}(y) = \begin{bmatrix} a_1 & a_2 & \ldots & a_J \\ a_2 & a_3 & \ldots & a_{J+1} \\ \vdots & \vdots & \ddots & \vdots \\ a_I & a_{I+1} & \ldots & a_L \end{bmatrix} \tag{2.3.16}$$

A Hankel tensor of order N, and therefore of size $I_1 I_2 ... I_N$, is obtained from a vector of length $\Sigma I_n - N + 1$ such that:

$$\chi(i_1, i_2, ..., i_N) = a(i_1 + i_2 + ... + i_N - N + 1) \tag{2.3.17}$$

It should be noted that any slice of the Hankel tensor (i.e. fixing $(N - 2)$ indices) are Hankel matrices. If it is the case that $I_n = I \forall n$ with identical dimensions, then it is also a symmetric tensor. A Hankel tensor can be constructed from another Hankel tensor of a smaller order by converting its fibers to Hankel matrices.

Convolution Tensor
Toeplitz and Hankel tensors enlarge the number of data entries of the original samples, and are therefore unsuitable for analysing signals of large sizes. Consider a third order tensor I×J×K where J=K-I+1, for which the $(l - I)^{th}$ diagonal elements of the l^{th} slice are all ones.

$$\chi(:, :, l) = \begin{bmatrix} 0 & 0 & 0 & \ldots & 0 \\ 1 & & \ddots & & \\ & \ddots & & & \\ 0 & \ldots & 1 & \ldots & 0 \end{bmatrix} \tag{2.3.18}$$

A further generalisation and properties are discussed in [23][24].

2.3.2 Decomposition Methods

This section will principally focus on three types of tensor decomposition: Canonical Polyadic (CP), Tucker and Tensor Train. An overview on the representation and algorithm of the methods will be provided, as well as their applications in data mining.

1. **Canonical Polyadic (CP) Decomposition**
   The Canonical Polyadic Decomposition (CPD), also known as PARAllel FACtor analysis (PARAFAC), eliminates the ambiguity associated with two-dimensional PCA and therefore has better uniqueness properties. It has been successfully applied in various domains such as chemometrics, telecommunications, psychometrics and data mining, to mention a few. Other applications include sensor array applications and Independent Component Analysis (ICA). For more details, refer to section 3.5 in [20] and references therein.

   A Polyadic Decomposition (PD) represents an Nth-order tensor $\chi \in R^{I_1 \times I_2 \times \ldots \times I_N}$ as a linear combination of rank-1 tensors in the form,

$$\boldsymbol{\chi} = \sum_{r=1}^{R} \lambda_r \boldsymbol{b}_r^{(1)} \circ \boldsymbol{b}_r^{(2)} \circ \cdots \circ \boldsymbol{b}_r^{(N)} \tag{2.3.19}$$

   Equivalently, χ is expressed as a multilinear product with a diagonal core:

$$\begin{aligned} \boldsymbol{\chi} &= \mathcal{D} \times {}_1\mathbf{B}^{(1)} \times {}_2\mathbf{B}^{(2)} \cdots \times_N \mathbf{B}^{(N)} \\ &= [[\mathcal{D}; \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(N)}]], \end{aligned} \tag{2.3.20}$$

   where $D = diag_N(\lambda_1, \lambda_2, ..., \lambda_R)$. Figure 2.3.5 illustrates these two interpretations for a third-order tensor. The tensor rank is defined as the smallest value of R for which Eq.(2.3.19) holds exactly. The minimum rank PD is called canonical (CPD) and is desired in signal separation.

   The matrix/vector form of CPD can be obtained via the Khatri-Rao products as:

$$\mathbf{X}_{(n)} = \mathbf{B}^{(n)} \mathbf{D} \left(\mathbf{B}^{(N)} \odot \cdots \odot \mathbf{B}^{(n+1)} \odot \mathbf{B}^{(n-1)} \odot \cdots \odot \mathbf{B}^{(1)} \right)^T \tag{2.3.21}$$

$$\text{vec}(\mathcal{X}) = \left[\mathbf{B}^{(N)} \odot \mathbf{B}^{(N-1)} \odot \cdots \odot \mathbf{B}^{(1)} \right] d \tag{2.3.22}$$

   Rank-related properties are very different for matrices and tensors. For instance, the number of complex-valued rank-1 terms needed to represent a higher-order tensor can be strictly less than the number of real-valued rank-1 terms, while the determination of tensor rank is in general NP-hard [25]. In signal processing applications, rank estimation often corresponds to determining the number of tensor components that can be retrieved with sufficient accuracy. Existing techniques for rank estimation include the CORCONDIA algorithm (CORe CONsistency DIAgnostic) which checks whether the core tensor is approximately diagonalizable [26], while there are a number of techniques that operate by balancing the approximation error versus the number of degrees of freedom for a varying number of rank-1 terms. Rank-R approximation of a d^{th}-order tensor $\chi \in R^{n_1 \times n_2 \ldots \times n_d}$ with $n_1 = n_2 = ... = n_d = n$ obtained by CPD is represented using O(Rdn) parameters, which is less than the number of parameters required for PCA applied to an unfolded matrix [27].

   Meaningful information lies in the factor matrices and hence, it is important to estimate these matrices from the data. Several algorithms were proposed for this purpose. The most popular is the Alternating Least Squares algorithm (ALS) which computes CPD by successively assuming the factors in d-1 modes known and then estimating the unknown set of parameters of the last mode [28]. For each mode and each iteration, the Frobenius norm of the difference between input tensor and CPD approximation is minimised. This iterative algorithm is very simple to implement and usually provides accurate results at sufficiently high SNR. However, it suffers from well-known convergence problems. In particular, the convergence is very sensitive to the initialisation and the existence of large amount of noise or the high order of the model may prevent ALS to converge to global minima or require thousands of iterations [3][6]. A second consequence is that it is difficult to efficiently set the threshold of the stopping criterion. Indeed, it frequently occurs that the algorithm escapes from a local minimum after a very large number of iterations and during these

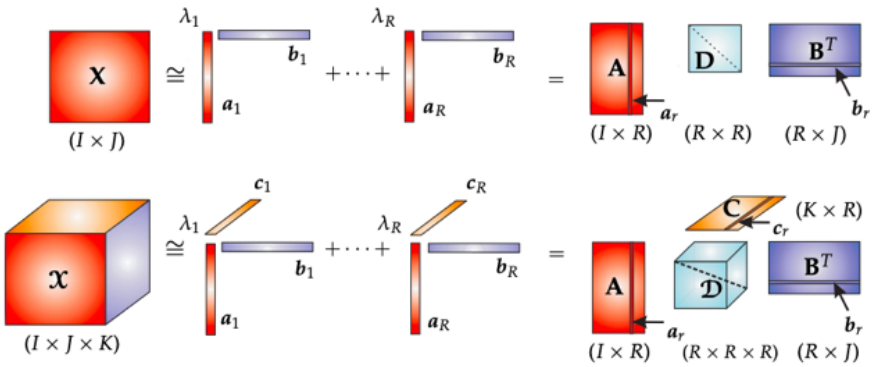

Figure 2.3.5: Analogy between dyadic (top) and polyadic (bottom) decompositions. Reprinted from "Multiway componentanalysis: Tensor decompositions for signal processing applications" by C. Caiafa A.-H. Phan G. Zhou Q. Zhao A. Cichocki, D. Mandic and L. D. Lathauwer., 2014, EEE Signal Processing Magazine, 51(3), p.7. Copyright by ArXiv

iterations the variations of the cost function can be very small. The decomposition performed with ALS can result in a high effective computational cost and thus can be time consuming when dealing with high rank CPD of large tensors. Several other iterative algorithms were proposed to solve these convergence problems but in practice, the computational cost of these solutions remains high. More details about ALS convergence problems and other iterative CPD algorithms can be found in [6][29][30][31]. Algorithm 1 shows the algorithm for CPD using ALS method.

Algorithm 1: CPD using ALS

Data: Tensor $\chi \in \mathbb{R}^{I_1 \times I_2 \ldots \times I_N}$, rank R

Result: $A^{(1)} \in \mathbb{R}^{I_1 xR}, A^{(2)} \in \mathbb{R}^{I_2 xR}, ..., A^{(N)} \in \mathbb{R}^{I_N xR}, \lambda \in \mathbb{R}^{1xR}$

(1) Initialise factor matrices $\mathbf{A}^{(n)} \in$ **repeat**

(2) **for** $n \leftarrow k$ **to** N **do**

(3) $V \leftarrow A^{(1)T} \cdot A^{(1)} * \cdots * A^{(n-1)T} \cdot A^{(m-1)} \cdot A^{(n+1)T} \cdot A^{(n+1)T} * \cdots * A^{(N)T} \cdot A^{(N)}$

(4) $A^{(n)} \leftarrow X^{(n)} \left(A^{(N)} \odot \cdots \odot A^{(n+1)} \odot A^{(n-1)} \odot \cdots \odot A^{(2)} \odot A^{(1)}\right) V^{\dagger}$

(5) $\lambda \leftarrow$ **Norm of** $\mathbf{A}^{(n)} columns$

(6) **Normalise** $\mathbf{A}^{(n)} columns$

(7) **end**

(8) **until** *convergence criteria met*;

(9) **return** $\lambda, A^{(N)}, A^{(N-1)}, ..., A^{(1)}$

2. **Tucker Decomposition**

An important issue with CP decomposition is that the rank N cannot be confirmed. If there are too many rank-1 tensors, the included information may be noise and redundant, otherwise this representation is incomplete. Therefore, it is difficult for CPD to effectively use the discriminative spatial information along each order [32]. Another decomposition strategy is the Tucker decomposition, which is considered as higher order Principal Component Analysis. For the Tucker decomposition, each tensor is represented as the product of a core tensor and factor matrices along all orders. Figure 2.3.6 illustrates the principle of Tucker decomposition which treats a tensor $\chi \in R^{I_1 \times I_2 \times \ldots \times I_N}$ as a multilinear transformation of a core tensor $G \in R^{R_1 \times R_2 \times \ldots \times R_N}$ by the factor matrices $B^{(n)} = [b_1^{(n)}, b_2^{(n)}, ..., b_{R_n}^{(n)}] \in R^{I_n \times R_n}, n = 1, 2, ..., N$, given by,

$$\boldsymbol{\chi} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_N=1}^{R_N} g_{r_1 r_2 \cdots r_N} \left(\boldsymbol{b}_{r_1}^{(1)} \circ \boldsymbol{b}_{r_2}^{(2)} \circ \cdots \circ \boldsymbol{b}_{r_N}^{(N)} \right) \tag{2.3.23}$$

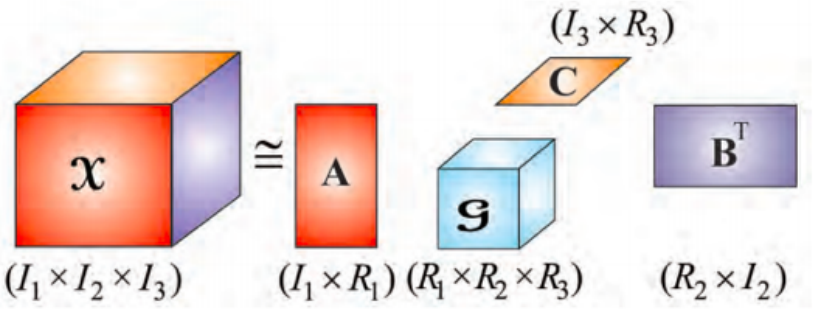

Figure 2.3.6: Tucker decomposition of a third-order tensor. The column spaces of A, B, C represent the signal subspaces for the three modes. The core tensor G is non-diagonal, accounting for possibly complex interactions among tensor components. Reprinted from "Multiway componentanalysis: Tensor decompositions for signal processing applications" by C. Caiafa A.-H. Phan G. Zhou Q. Zhao A. Cichocki, D. Mandic and L. D. Lathauwer., 2014, EEE Signal Processing Magazine, 51(3), p.10. Copyright by ArXiv

or equivalently,

$$\begin{aligned} \boldsymbol{\chi} &= \boldsymbol{g} \times \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \cdots \times_N \mathbf{B}^{(N)} \\ &= \mathbf{G}; \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(N)} \end{aligned} \tag{2.3.24}$$

Using the Kronecker products, Tucker decomposition can be expressed in a matrix/vector form as:

$$\mathbf{X}_{(n)} = \mathbf{B}^{(n)} \mathbf{G}_{(n)} \left(\mathbf{B}^{(N)} \otimes \cdots \otimes \mathbf{B}^{(n+1)} \otimes \mathbf{B}^{(n-1)} \otimes \cdots \otimes \mathbf{B}^{(1)} \right)^T \tag{2.3.25}$$

$$\text{vec}(\mathcal{X}) = \left[\mathbf{B}^{(N)} \otimes \mathbf{B}^{(N-1)} \otimes \cdots \otimes \mathbf{B}^{(1)} \right] \text{vec}(\mathbf{G}) \tag{2.3.26}$$

For a core tensor of minimal size, R_1 is the column rank (the dimension of the subspace spanned by mode-1 fibers), R_2 is the row rank (the dimension of the subspace spanned by mode-2 fibers), and so on. A remarkable difference from matrices is that the values of $R_1, R_2, ..., R_N$ can be different for N $\geqslant$ 3. The N-tuple $(R_1, R_2, ..., R_N)$ is consequently called the multilinear rank of the tensor χ [21]. Rank-R approximation of a d^{th}-order tensor $\chi \in R^{n_1 \times n_2 \ldots \times n_d}$ with $n_1 = n_2 = ... = n_d = n$ obtained by Tucker Decomposition is represented using $\mathbf{O}(Rdn + R^d)$ parameters.

Eq.(2.3.23) shows that Tucker decomposition can be considered as an expansion in rank-1 terms, while Eq.(2.3.20) represents CPD as a multilinear product of a core tensor and factor matrices (but the core is not necessary minimal). However, despite the obvious interchangeability of notation, the CP and Tucker decompositions serve different purposes. In general, the Tucker core cannot be diagonalized, while the number of CPD terms may not be bounded by the multilinear rank. Consequently, in signal processing and data analysis, CPD is typically used for factorising data into easy to interpret components (i.e., the rank-1 terms), while the goal of unconstrained Tucker decomposition is most often to compress data into a tensor of smaller size (i.e., the core tensor) or to find the subspaces spanned by the fibers (i.e., the column spaces of the factor matrices). Also, unlike CPD, Tucker decomposition is not unique. However, constraints such as orthogonality, non-negativeness, sparsity, independence and smoothness have been imposed on the factor matrices to obtain unique decompositions. [21]

Tucker Decomposition primarily includes two algorithms: Higher Order Singular Value Decomposition (HOSVD) and High Order Orthogonal Interaction (HOOI).

The Higher Order SVD (HOSVD) [33] is a special case of Tucker decomposition obtained by adding an orthogonality constraint to the component matrices. The algorithm for HOSVD, as shown in Algorithm 2, is based on the following idea: [19]

- First unfold X along mode k to get X(k);
- Then, perform a singular value decomposition (SVD): $X(k) = U_k S_k V_k^T$;

- Finally, pick B_k as the first R_k columns of U_k.

Therefore, in HOSVD, low n-rank approximation of χ is obtained by truncating the orthogonal factor matrices $U^{(i)}s$ of HOSVD resulting in truncated HOSVD. Due to the orthogonality of the core tensor, HOSVD is unique for a specific multilinear rank.

Algorithm 2: Higher-Order Singular Value Decomposition

Data: Tensor $\chi \in \mathbb{R}^{I_1 \times I_2 \ldots \times I_N}$, ranks $R_1, ..., R_N$
Result: $U_1 \in \mathbb{R}^{I_1 x R}, U_2 \in \mathbb{R}^{I_2 x R}, ..., U_N \in \mathbb{R}^{I_N x R}, \mathcal{K} \in \mathbb{R}^{R_1 x \ldots x R_N}$
(1) **for** $n \leftarrow 1$ **to** N **do**
(2) $\quad [\boldsymbol{U}, \sigma, \boldsymbol{V}] \leftarrow SVD\left(X_{(n)}\right)$
(3) $\quad \boldsymbol{U}_n \leftarrow \boldsymbol{U}\left(:, 1 : R_n\right)$
(4) **end**
(5) $\mathcal{K} \leftarrow \chi \times_N U_N^T \times_{N-1} \cdots \times_1 U_1^T$
(6) **return** $U_1, U_2, ..., U_N$ *and* $\mathcal{K}$

Although this method is easy to implement, it is not optimal in fitting the data. Alternatively, an Alternative Least-Square (ALS) method called HOOI is widely used to get a better solution.

The High Order Orthogonal Iteration (HOOI) [34][35][36] method aims to minimize the approximation error:

$$\min_{\mathbf{G}, \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}} \left\| \boldsymbol{\chi} - \mathbf{G}; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)} \right\| \tag{2.3.27}$$

subject to $\mathbf{G} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$
$\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ and columnwise orthogonal for $n = 1, \ldots, N$
through the iterative process as shown in Algorithm 3. Each iteration of the HOOI consists of two steps for every mode k:

- Obtain a tensor B via a power iteration (n-mode (matrix) product along all modes except k), which can be done from mode d, then d1, ... until mode 1.
- Perform an SVD of the mode-k unfolded matrix of B to extract a mode-k factor matrix A_k.

In practice, the initialisation process via HOSVD can be very time-consuming, because it needs d SVD operations, and each of it works on a matrix whose size is equal to the original tensor. Therefore, some random orthonormal matrices are often used as the initial factor matrices. In this case, the total number of iterations needed may increase slightly, but the total runtime can decrease significantly. Even though, when the size of the tensor χ is large, the time and energy consumed to compute the Tucker decomposition can be very high.

Algorithm 3: Higher-Order Orthogonal Iteration

Data: Tensor $\chi \in \mathbb{R}^{I_1 \times I_2 \ldots \times I_N}$, ranks $R_1, ..., R_N$
Result: $U_1 \in \mathbb{R}^{I_1 x R}, U_2 \in \mathbb{R}^{I_2 x R}, ..., U_N \in \mathbb{R}^{I_N x R}, \mathcal{K} \in \mathbb{R}^{R_1 x \ldots x R_N}$
(1) Initialise $U_1 ... U_N$ using HOSVD or randomly
(2) **repeat**
(3) $\quad$ **for** $n \leftarrow 1$ **to** N **do**
(4) $\quad\quad \mathcal{W} \leftarrow x \times_N U_N^T \cdots \times_{n-1} \boldsymbol{U}_{n-1}^T \times_{n+1} \boldsymbol{U}_{n+1}^T \cdots \times_1 \boldsymbol{U}_1^T$
(5) $\quad\quad [\boldsymbol{U}, \Sigma, V] \leftarrow SVD\left(\boldsymbol{W}_{(n)}\right)$
(6) $\quad\quad \boldsymbol{U}_n \leftarrow \boldsymbol{U}\left(:, 1 : R_n\right)$
(7) $\quad$ **end**
(8) **until** *convergence criteria met*;
(9) $\mathcal{K} \leftarrow \chi \times_N U_N^T \times_{N-1} \cdots \times_1 U_1^T$
(10) **return** $U_1, U_2, ..., U_N$ *and* $\mathcal{K}$

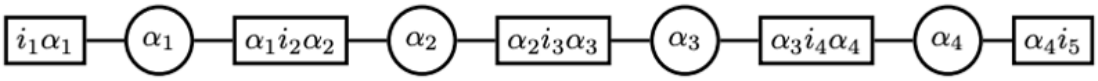

Figure 2.3.7: Tensor Train Network. Reprinted from "Tensor-train decomposition" by I. V. Oseledets, 2011, SIAM Journal on Scientific Computing, 33(5), p.2297, Copyright by SIAM

Tucker decomposition has various applications in signal processing [3][37], image processing [38], data mining and machine learning [39][40], pattern recognition [41], computer vision [42][43] and chemical analysis [44][45]. It is a fundamental decomposition technique for other methods, such as hierarchical Tucker decomposition and Tensor Train decomposition. As with many decomposition methods, it can be used for dimensionality reduction in order to compute CPD with better efficiency.

3. **Tensor Train Decomposition**
Unlike decompositions methods such as PARAFAC and Tucker that decompose complex high dimensional data tensors into their factor tensors and matrices, Tensor networks (TN) represent a higher-order tensor as a set of sparsely interconnected lower-order tensors, typically 3^{rd}-order and 4^{th}-order tensors called core, and provide computational and storage benefits [4][46][47]. There are several tensor network topologies but in this report, we will discuss only Tensor Train decomposition.

Tensor Train (TT) Decomposition can be interpreted as a special case of the Hierarchical Tensor (HT) decomposition. HT decomposition reduces the memory requirements of Tucker decomposition by recursively splits the modes based on a hierarchy and creates a binary tree containing a subset of the modes t $\subset$ [d] at each node [48][49][50].

In TT decomposition, all nodes of the underlying tensor network are connected in a cascade or train. It has been proposed to compress large tensor data into smaller core tensors. When the complexity of data increases, this model allows users to avoid the exponential growth of Tucker model and provides more efficient storage complexity. TT decomposition of a tensor $\chi \in R^{n_1 \times n_2 \times \ldots \times n_d}$ is written as:

$$\chi_{\mathrm{i}_1,\ldots \mathrm{i}_\mathrm{d}} = \mathrm{G}_1\left(\mathrm{i}_1\right) \cdot \mathrm{G}_2\left(\mathrm{i}_2\right) \cdots \mathrm{G}_\mathrm{d}\left(\mathrm{i}_\mathrm{d}\right), \tag{2.3.28}$$

where $G_k(i_k) \in R^{R_k 1 \times R_k}$ is the i_k^{th} lateral slice of the three-dimensional core $G_k \in R^{R_k 1 \times n_k \times R_k}$ (analogous to the core of the Tucker decomposition), R_k being the TT-rank with $R_0 = R_d = 1$.
We can write Eq.(3.28) in the index form. Matrix $G_k(i_k)$ is actually a three-dimensional array, and it can be treated as an $r_k 1 \times n_k \times r_k$ array with elements $G_k(\alpha_k 1, n_k, \alpha_k) = G_k(i_k)(\alpha_k 1)\alpha_k$. In the index form the decomposition is written as,

$$A\left(i_1, \ldots, i_d\right) = \sum_{\alpha_0,\ldots,\alpha_{d-1},\alpha_d} G_1\left(\alpha_0, i_1, \alpha_1\right) G_2\left(\alpha_1, i_2, \alpha_2\right) \ldots G_d\left(\alpha_{d-1}, i_d, \alpha_d\right) \tag{2.3.29}$$

Since $R_0 = R_d = 1$ this decomposition can also be represented graphically by a linear tensor network, which is presented in Figure 2.3.7 for d = 5. In the network, there are two types of nodes. Rectangles contain spatial indices (i.e., the indices i_k of the original tensor) and some auxiliary indices α_k, and a tensor with these indices is associated with such kind of nodes. Circles contain only the auxiliary indices k and represent a link. This means if an auxiliary index is present in two cores, we connect it. The summation over the auxiliary indices is assumed; i.e., to evaluate an entry of a tensor, one has to multiply all tensors in the rectangles and then perform the summation over all auxiliary indices. This picture looks like a train with carriages and links between them, and that justifies the name Tensor Train decomposition. The storage complexity of TT-rank-R approximation of a d^{th}-order tensor $\chi \in R^{n_1 n_2 \ldots n_d}$ with $n_1 = n_2 = \ldots = n_d = n$ is $\mathrm{O}(dnR^2)$. [51]

The simplest method of computing the tensor train decomposition is using the TT-SVD algorithm which involves a sequence of SVD computations on reshaped matrices. It has been described in [51].
The advantages of the TT model over HT are:

- Its simpler practical implementation as no binary tree needs to be determined

- The simplicity of the computation
- Computational efficiency (linear in the tensor order)

Although TT format has been used widely in linear equation solution [52], electronic design automation (EDA) [53][54][55], system identification [56], large-scale matrix processing [57][58][59], signal processing [3], machine learning [60] and chemical analysis [61], it suffers from a couple of limitations. TT model requires rank-1 constraints on the border factors, i.e. they have to be matrices. And most importantly, the multiplications of the TT cores are not permutation invariant, requiring the optimisation of the ordering using procedures such as mutual information estimation. These drawbacks have been recently addressed by the tensor ring (TR) decomposition [62][63]. TR decomposition removes the unit rank constraints for the boundary cores and utilises a trace operation in the decomposition, which removes the dependency on the core order.

2.4 Financial Terminologies

2.4.0.1 Asset Classes

An asset class is a grouping of investments that exhibit similar characteristics and are subject to the same laws and regulations. Asset classes are made up of instruments which often behave similarly to one another in the marketplace. There is usually very little correlation between different asset classes. Historically, there are three asset classes: [64]

- **Stock or Equities**: Equities are shares of ownership issued by public-trades companies. They are traded on stock exchanges such as NYSE or NASDAQ. Companies or individuals can profit from equities ether through a rise in share prices or by receiving dividends.
- **Bonds or other fixed-income investment**s: Fixed income investments are investments in debt securities that pay a rate of return in the form of interest.
- **Cash and cash equivalents, such as money market funds**: The primary advantage of cash or cash equivalent investments is their liquidity. Money help in the form of cash can be easily accessed at any time.

Currently, many investment professionals also include illiquid assets such as real estate, commodities and other tangible assets, other financial derivatives, and even cryptocurrencies to the asset class. Investment assets include both tangible and intangible instruments which investors buy and sell for the purposes of generating additional income on either a short- or a long-term basis.

Asset classes are a way to help investors diversify their portfolio. Different asset classes have different cash flows streams and varying degrees of risk. Investing in several different asset classes ensures a certain amount of diversity in investment selections. Diversification reduces risk and increases your probability of making a return [65].

The only asset this project focuses on are stocks, but general principles apply to all kinds of assets. Overall, the financial markets produce market data such as pricing and volume information across a range of asset classes which exhibit structural relationships, hence making it ideal for tensor processing methods.

2.4.1 Portfolio

A portfolio is a collection of multiple financial assets, and is characterised by its: [66]

- **Constituents**: M assets of which it consists;
- **Portfolio vector,** w_t: its i^{th} component represents the ratio of the total budget invested to the i^{th} asset, such that:

$$w_t = \begin{bmatrix} w_{1,t}, & w_{2,t}, \ldots, & w_{M,t} \end{bmatrix}^T \in \mathbb{R}^M \text{ and } \sum_{i=1}^{M} w_{i,t} = 1 \tag{2.4.1}$$

For fixed constituents and portfolio vector w_t, a portfolio can be treated as a single master asset. Therefore, the analysis of single simple assets can be applied to portfolios upon determination of the constituents and the corresponding portfolio vector. Portfolios are more powerful, general representation of financial assets since the single asset case can be represented by a portfolio, where the j^{th} asset is equivalent to the portfolio with vector $e^{(j)}$, and the j^{th} term is equal to unity and the rest are zero. Portfolios are also preferred over single assets in order to minimise risk.

2.4.2 Asset Price

Let $p_t \in$ be the price of an asset at discrete time index t, then the sequence $p_1, p_2, ..., p_T$ is a univariate time-series. The T-samples price time-series of an asset i, is the column vector $p_{i,1:T}$, such that: [66]

$$\vec{p}_{i,1:T} = \begin{bmatrix} p_{i,1} \\ p_{i,2} \\ \vdots \\ p_{i,T} \end{bmatrix} \in \mathbb{R}_+^T \tag{2.4.2}$$

Extending the single-asset time-series notation to the multivariate case, the asset price matrix $P_{1:T}$ is formed by stacking column-wise the T samples price time-series of the M assets of the portfolio, then:

$$\overrightarrow{\boldsymbol{P}}_{1:T} = \begin{bmatrix} \vec{p}_{1,1:T'} & \vec{p}_{2,1:T'} & \dots, & \vec{p}_{M,1:T} \end{bmatrix} = \begin{bmatrix} p_{1,1} & p_{2,1} & \cdots & p_{M,1} \\ p_{1,2} & p_{2,2} & \cdots & p_{M,2} \\ \vdots & \vdots & \ddots & \vdots \\ p_{1,T} & p_{2,T} & \cdots & p_{M,T} \end{bmatrix} \in \mathbb{R}_+^{T \times M} \tag{2.4.3}$$

2.4.3 Asset Returns

Price changes over time reflects the investment profit and loss, which is called return from the asset. There are three types of returns: [66]

- **Gross returns**: The gross return R_t of an asset represents the scaling factor of an investment in the asset at time (t-1). It is given by the ratio of its prices at times t and (t-1), such that:

$$R_t \triangleq \frac{p_t}{p_{t-1}} \in \mathbb{R} \tag{2.4.4}$$

- **Simple returns**: The simple return, r_t, which represents the percentage change in asset price from time (t-1) to time t, such that:

$$r_t \triangleq \frac{p_t - p_{t-1}}{p_{t-1}} = \frac{p_t}{p_{t-1}} - 1 \overset{(2.4.4)}{=} R_t - 1 \in \mathbb{R} \tag{2.4.5}$$

  The **portfolio simple return** can be represented as the linear combination of the simple returns of each constituent, weighted by the portfolio vector. Hence, at time index t, we obtain:

$$r_t \triangleq \sum_{i=1}^{M} w_{i,t} r_{i,t} = w_t^T r_t \in \mathbb{R} \tag{2.4.6}$$

- **Log returns**: Despite the interpretability of the simple return as the percentage change in asset price over one period, it is asymmetric and therefore practitioners tend to use log returns instead, in order to preserve interpretation and to yield a symmetric measure. The log return ρ_t at time t is:

$$\begin{gathered} \rho_t \triangleq \ln\left(\frac{p_t}{p_{t-1}}\right) \overset{(2.4.4)}{=} \ln\left(R_t\right) \in \mathbb{R} \\ \rho_t = \ln\left(1 + r_t\right) \end{gathered} \tag{2.4.7}$$

  Moreover, since gross return is centred around unity, the logarithmic operator makes log returns concentrated around zero. By replacing r_t in the previous formula, we get the portfolio log return:

$$\rho_t \triangleq \ln\left(1 + w_t^T r_t\right) \in \mathbb{R} \tag{2.4.8}$$

2.4.4 Evaluation Criteria

A set of evaluation criteria and metrics is necessary in order to evaluate the performance of the generated portfolios. Due to the uncertainty of the future dynamics of the financial markets, we study the statistical properties of the assets returns, as well as risk and performance metrics. [66]

- **Statistical Metrics:**
  - **Mean**: The weighted average of each possible outcome (expected returns in this case), where the weight of each outcome is provided by its respective probability, is called the **expected value** or **mean** of the random variable. It is given by:

$$\mathbb{E}[X] = \mu_X \triangleq \int_{-\infty}^{+\infty} x f_X(x) dx \in \mathbb{R} \tag{2.4.9}$$

    where f_x is the probability distribution function. For the same level of risk, one should choose the portfolio that maximises the expected returns.

  - **Volatility and Covariance**: The variance measures how far the random variable X is spread out of its mean, and is given by:

$$\mathrm{Var}[X] = \sigma_X^2 \triangleq \mathbb{E}\left[(X - \mathbb{E}[X])^2\right] \in \mathbb{R} \tag{2.4.10}$$

    The square root of the variance, X, namely the **standard deviation** or **volatility** in finance, is the measure of dispersion of returns and is a more physically interpretable parameter, since it has the same units as the random variable under consideration (i.e., prices, simple returns). For the same expected returns, one should choose the portfolio that minimizes the volatility.
    The variance is extended to the multivariate case by introducing covariance. It measures the joint variability of two variables and is given by:

$$\mathrm{Cov}\left[X_m, X_n\right] = [\mathrm{Cov}[\mathbf{X}]]_{mn} = \Sigma_{mn} \triangleq \mathbb{E}\left[\left(X_m - \mathbb{E}\left[X_m\right]\right)\left(X_n - \mathbb{E}\left[X_n\right]\right)\right] \in \mathbb{R} \tag{2.4.11}$$

    The correlation coefficient is also frequently used to quantify the linear dependency between random variables. It takes values in the range [1, 1] and is given by:

$$\mathrm{corr}\left[X_m, X_n\right] = [\mathrm{corr}[\mathbf{X}]]_{mn} = \rho_{mn} \triangleq \frac{\mathrm{Cov}\left[X_m, X_n\right]}{\sigma_{X_m}\sigma_{X_n}} \in [-1, 1] \subset \mathbb{R} \tag{2.4.12}$$

  - **Skewness**: It is the standard measure of symmetry of a distribution and is given by:

$$\mathrm{skew}[X] \triangleq \frac{\mathbb{E}\left[(X - \mathbb{E}[X])^3\right]}{\sigma_X^3} \tag{2.4.13}$$

    A distribution whose probability density function is symmetric around its expected value has null skewness. If the skewness is positive (negative), occurrences larger than the expected value are less (more) likely than occurrences smaller than the expected value. Hence, one should choose negatively skewed returns, rather than positively skewed. It is characterized by many small gains and a few extreme losses.

  - **Kurtosis**: It is a measure of the relative weight of the tails with respect to the central body of a distribution. a large kurtosis implies that the distribution displays "fat tails".

$$\mathrm{kurt}[X] \triangleq \frac{\mathbb{E}\left[(X - \mathbb{E}[X])^4\right]}{\sigma_X^4} \tag{2.4.14}$$

    Low kurtosis is usually preferred in a distribution such that most of the returns are not far away from the mean.

- **Risk and Performance Metrics:**
  - **Cumulative Returns**: It represents the change in asset prices over larger time horizons. The cumulative gross return, $R_{t\to T}$ between time indexes t and T is given by:

$$R_{t\to T} \triangleq \frac{p_T}{p_t} = \left(\frac{p_T}{p_{T-1}}\right)\left(\frac{p_{T-1}}{p_{T-2}}\right)\cdots\left(\frac{p_{t+1}}{p_t}\right) \overset{(2.4.4)}{=} R_T R_{T-1}\cdots R_{t+1} = \prod_{i=t+1}^{T} R_i \in \mathbb{R} \tag{2.4.15}$$

  Similarly, cumulative simple return, $r_{t\to T}$, and cumulative log return, $\rho_{t\to T}$ are given by:

$$r_{t\to T} \triangleq \tfrac{p_T}{p_t} - 1 \overset{(2.4.15)}{=} \left[\textstyle\prod_{i=t+1}^{T} R_i - 1\right] \overset{(2.4.5)}{=} \left[\textstyle\prod_{i=t+1}^{T} (1 + r_i) - 1\right] \in \mathbb{R} \tag{2.4.16}$$

$$\rho_{t\to T} \triangleq \ln\left(\frac{p_T}{p_t}\right) \overset{(2.4.15)}{=} \ln\left(\prod_{i=t+1}^{T} R_i\right) = \sum_{i=t+1}^{T} \ln(R_i) \overset{(2.4.7)}{=} \sum_{i=t+1}^{T} \rho_i \in \mathbb{R} \tag{2.4.17}$$

  One should aim to maximise profitability of investment.

  - **Sharpe Ratio**: It is the ratio of expected returns to their standard deviation, adjusted by a scaling factor, which is equivalent to **Signal to Noise ratio (SNR)**:

$$\mathbf{SR}_{1:T} \triangleq \sqrt{T}\frac{\mathbb{E}[\mathbf{r}_{1:T}]}{\sqrt{\mathrm{Var}[\mathbf{r}_{1:T}]}} \in \mathbb{R} \tag{2.4.18}$$

  One should aim to maximise Sharpe Ratio of investment.

  - **Maximum Drawdown**: It is the measure of the maximum decline from a historical peak in cumulative returns. is usually quoted as the percentage between the peak and the subsequent trough and is defined as:

$$\mathrm{MDD}(t) = -\max_{x\in(0,t)}\left\{\left[\max_{\tau\in(0,T)} r_{0\to\tau}\right] - r_{0\to T}\right\} \tag{2.4.19}$$

Preference for the metrics used for performance evaluation is defined in Table 2.2.

Metric	Preference
Annual return	High
Cumulative returns	High
Annual volatility	High
Sharpe ratio	High
Maximum drawdown	Low
Skew	High
Kurtosis	Low

Table 2.4.1: Performance metrics and their preference

2.4.5 Factor Models

Investment portfolios are typically constructed using asset classes as building blocks. By combining multiple asset classes in a portfolio, managers try to ensure risk diversification. However, in reality asset classes often share significant overlapping risk exposures. Constructing portfolios using factor based investing, apart from asset classes, can guide managers to potentially build more efficient portfolios that require less risk to achieve competitive returns. Factor models help explain the types of risk in a portfolio and quantify how much of each risk is present in the portfolio. They are also used to create risk forecasts that help portfolio managers to make de-risking and allocation decisions. Managers can also incorporate the most relevant model-derived factors in order to construct more realistic and robust portfolios. [67]
There are three kinds of factor models:

- **Fundamental**: This approach is based on decades of academic research into asset price behavior. Fundamental models decompose risk using equity and fixed income factors such as company size, market capitalization, interest rate, credit and repayment risk and balance sheet information. Fundamental models are ideal for performance attribution and portfolio risk decomposition. However, since fundamental models are defined in terms of known factors, they can become less accurate when new, unknown factors emerge.

- **Statistical**: Statistical models use a numerical technique called Principal Component Analysis (PCA), to calculate risk exposures, without requiring any assumptions about what those factors represent. PCA based models adapt readily to changing trends and correlations in the underlying markets, unlike fundamental models that utilize fixed factors. However, the factors derived from statistical models lack the easy interpretability of fundamental models.

- **Macroeconomic**: Macroeconomic model calculates risk exposures using observable economic time series such as inflation, industrial production, and the U.S. Dollar exchange rate as factors. This model augments fundamental and statistical factor models by gauging the impact of economic conditions on a portfolio. Macroeconomic models are primarily used for performance (return-based) attribution.

2.5 Financial Implementation

The financial implementation includes two methods:

1. Statistical factor analysis

2. Tensor decomposition methods

For statistical factor analysis, the conventional PCA and ICA are applied on the stocks returns data to compute weights for the assets.

The goal of the second method is to combine data for asset returns and fundamental factors together using tensor algebra and perform decomposition to construct a portfolio using the assets. Generally, numerical techniques such as Principal Component Analysis (PCA) and Independent Component Analysis (ICA) are used to find the components, that represent statistical factors. A new approach based on tensorizing the stock returns and fundamental factors data and applying tensor decomposition methods, as discussed in section 2.3.2, to decompose the data into lower-rank matrices and applying statistical methods on them to construct the portfolio. HottBox [68], a tensor toolbox, is used to tensorise our stock returns, volume and fundamental factors data into a three-dimensional tensor T^{IxJxK}, where mode I is the time step (dates), mode J is the number of stocks and mode K is the number of types of stocks data used.

Financial fundamental factors used in our analysis include:

- **Momentum**: Momentum is the rate of acceleration of an asset's price, i.e., the speed at which the price is changing. It is the observed tendency of financial assets trending strongly in a certain direction (rising or falling) to continue to move in that direction. Market momentum is measured by continually taking price differences for a fixed time interval. For our analysis, the 10-day momentum is calculated by simply subtracting the closing price 10 days ago from the last closing price. [69]

- **Price to Book (P/B) Ratio**: The ratio is used to compare a stock's market value to its book value. When comparing the P/B ratios between similar companies, with all other factors being equal, the one with a lower P/B ratio is more attractive. It is calculated as follows:
  P/BV Ratio = Current Market Price per Share / Book Value per Share [70]

- **Price to Equity (P/E) Ratio**: The ratio is commonly used to know valuation of a share of the company. It indicates how much investors are paying in reference to the company's earnings. P/E ratio between companies in the same industry can be compared to determine undervalued and overvalued stocks. With all other factors being equal, assets with a lower P/E ratio are more attractive. It is calculated as follows:
  P/E = Price per Share / Earnings per Share [70]

- **Market Capitalisation**: Market capitalisation measures the total value of a company by taking the total market value of all of its outstanding shares. It's calculated by multiplying the current market price of its stock by the number of shares outstanding. [71]
- **Non-Predictive Open Price Factor**: Non-Predictive factor is similar to momentum, but it is calculated using open prices instead of closing prices. The performance of the stocks is evaluated on their trends in the last 10 days.
- **Predictive Open Price Factor**: Predictive factor works on look-ahead bias where information or data used in a study or simulation is not known or available during the period analyzed. It is calculated here using the future prices for the next 5 days.

These six fundamental factors are used for the purpose of analysis in this project. However, other fundamental factors including, enterprise value, price-to-sales ratio, debt-to-equity ratio, return-on-equity, profit margin, net profit, etc. can be used as well. [70][71]

CHAPTER 3

Data Selection and Pre-processing

3.1 Data Selection

Before beginning with the analysis, it is important to choose a small number of assets from a single asset class. For the purpose of this project, 20 leading US companies belonging to the Technology asset class are considered, which are listed on the New York Stock Exchange (NYSE). The stocks are selected such that the stocks market data can be accessed from Yahoo finance and the fundamental financial data can be accessed from SimFin. Table 3.1.1 shows all the selected stocks. Technology asset class was chosen in particular due to the easy availability of financial data for these stocks on various platforms. The stocks data used for the analysis include market data, such as adjusted closing prices, open prices and volume, and fundamentals data, such as market capital, Price to Book ratio and Price to Equity ratio.

3.2 Data Pre-processing

The data of the above assets were obtained from Yahoo Finance and contain the daily opening and adjusted closing prices and volume from May 20, 2011 to May 20, 2021. Raw data for Alphabet, NVIDIA Corporation, Advanced Micro Devices and Visa are presented in Figure 3.2.1. The raw data plots for remaining assets are provided in Appendix C.

One of the aspects that needs to be considered for financial analysis is that the time series should be stationary. A stationary signal has constant mean and variance over time. Moreover, the autocorrelation also remains the same over time. Autocorrelation is the situation in which the time series data is influenced by historical values. It is important to stationarize a time series because most statistical models are based on the assumption that the underlying time series can be transformed into stationary series using mathematical transformations. However, for most assets, the price data does not exhibit stationary behaviour, as observed in Figure 3.2.1 where most asset prices rise without bound. As a result, in quantitative finance, the pricing data is ideally transformed into log returns through differencing the log-transform of prices using Eq(2.4.7). The benefit of using returns over prices is normalisation, which is a requirement for several multidimensional statistical analysis and machine learning techniques. [72] If prices are assumed to be log-normally distributed, returns tend to be normally distributed in the short-run. However, log returns have some undesirable properties. They do not give a direct measure of the change in wealth and the upward trend of the market tends to skew the distribution in the long run, which are explained in [73].

The stationarity of log returns for Alphabet, NVIDIA Corporation, Advanced Micro Devices and Visa are demonstrated in Figure 3.2.2. The log-returns plots for remaining assets are provided in Appendix C.

Assets	Tickers
Apple	AAPL
Adobe	ADBE
Applied Materials	AMAT
Advanced Micro Devices	AMD
Amazon	AMZN
Cisco Systems	CSCO
Alphabet	GOOGL
Hewlett-Packard	HPQ
IBM	IBM
Intel Corporation	INTC
Microsoft	MSFT
Netflix	NFLX
NVIDIA Corporation	NVDA
Qualcomm	QCOM
AT&T	T
Tesla	TSLA
Texas Instruments	TXN
Visa	V
Verizon Communications	VZ
Xilinx	XLNX

Table 3.1.1: Selected stocks

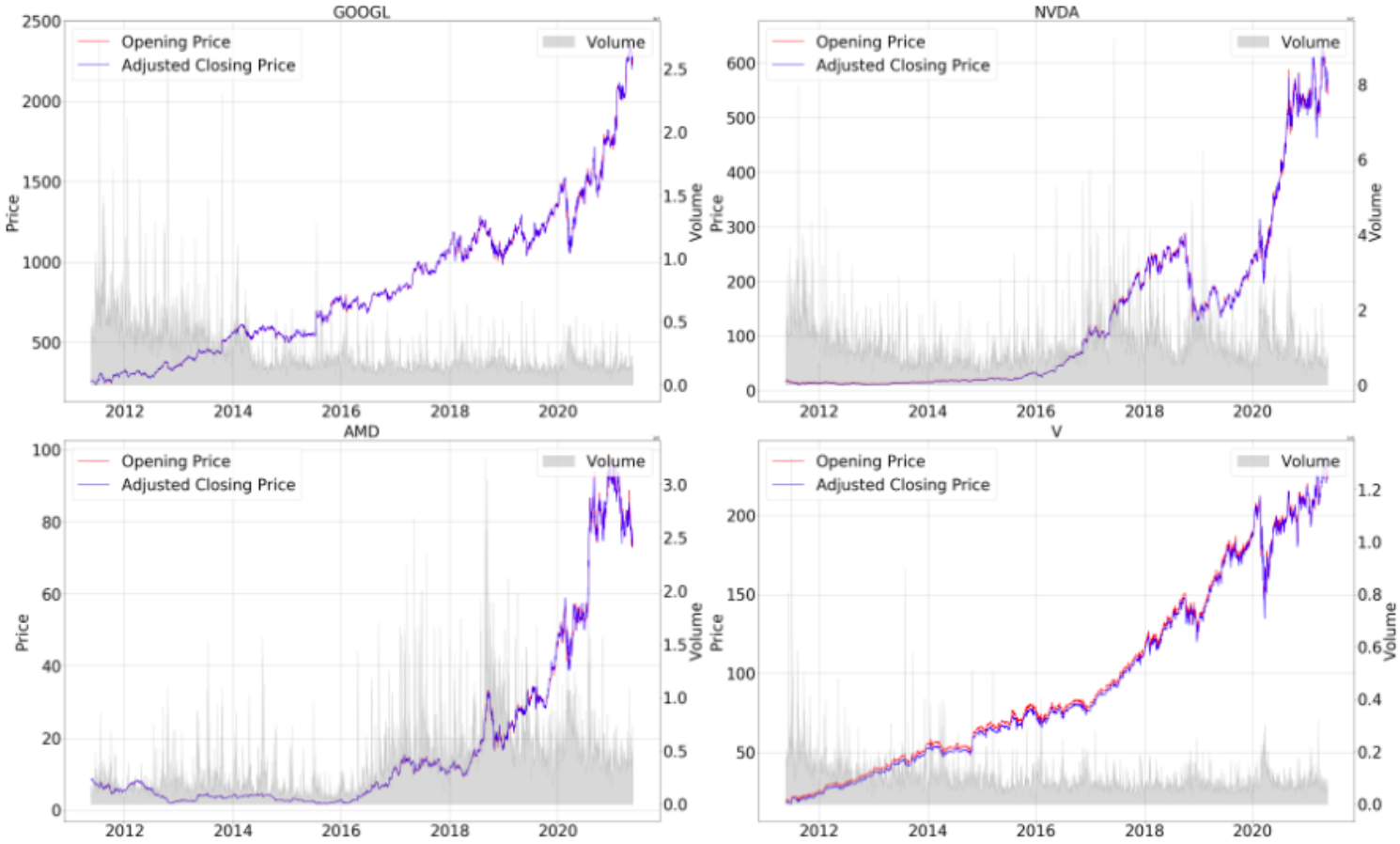

Figure 3.2.1: Raw price and volume data for GOOGL, NVDA, AMD, V Stocks

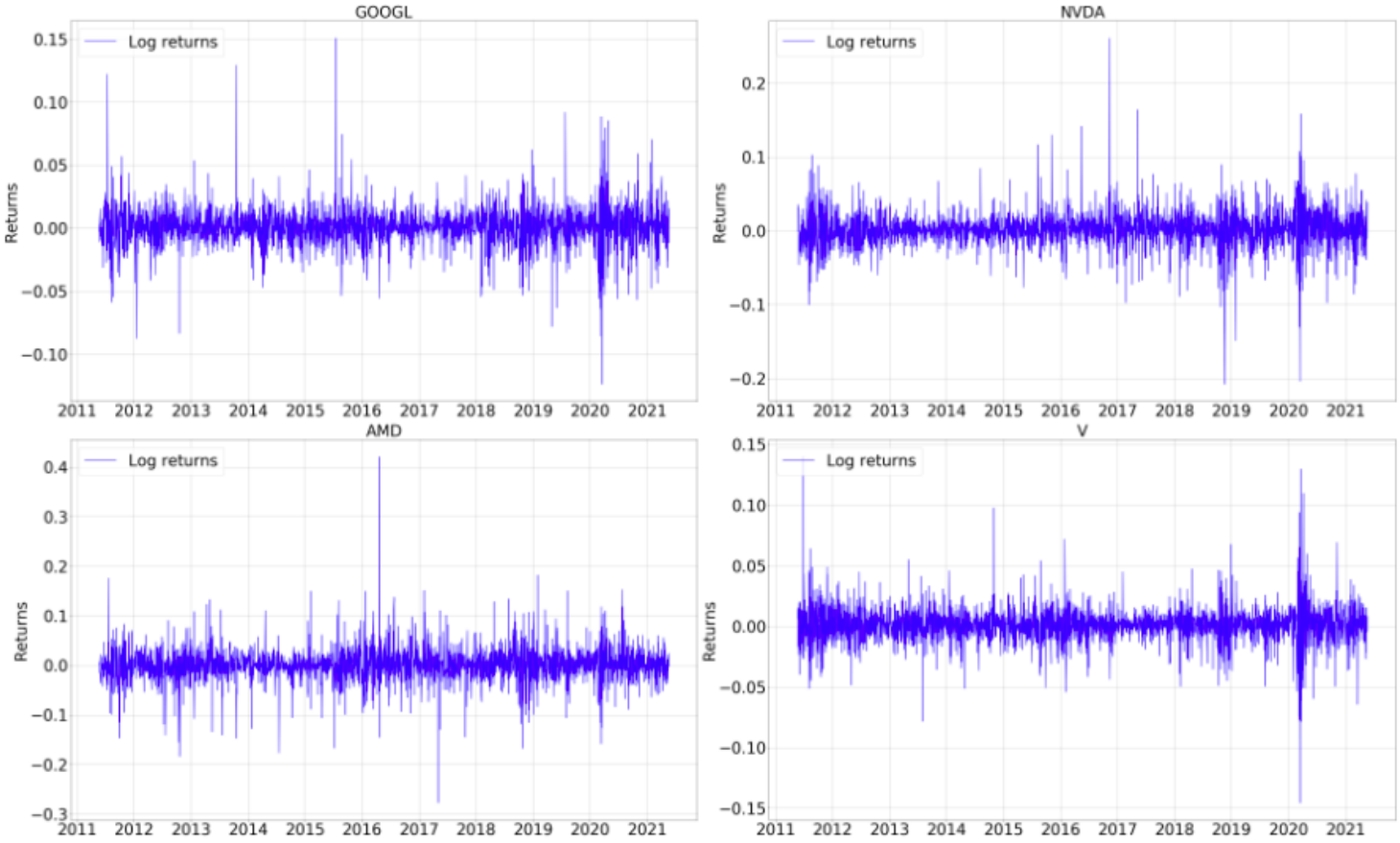

Figure 3.2.2: Log-returns for GOOGL, NVDA, AMD, V Stocks

The stocks data is divided into training and testing set in the ratio 80:20. Therefore, data for the first 8 years, that is, between May 20, 2011 to May 20, 2019 is used for finding the dominant assets and asset weights, and the 2 years' data from May 21, 2019 to May 20, 2021 is used for testing the dominant assets and weights obtained from training.

For Tensor-based analysis, stocks market data and fundamentals data are combined together to form a third-order tensor $T \in R^{2011x20x8}$, with the modes corresponding to 2011 time steps/dates, 20 assets and 8 raw features (returns, volume and financial fundamentals data).

CHAPTER 4

Statistical Factor Analysis on Stock Returns Data

4.1 Principal Component Analysis

Principal Component Analysis decomposes the data in several vectors known as principal components. These principal components are a linear combinations of input features and are ordered by the amount a variance explained by each vector, with the first principal component explaining most of the information. In context to portfolio construction, PCA can be applied to daily stock returns to find the principal components, which is a vector containing weights for each of stocks. The first principal component is considered as it is the primary driver of stock returns, explaining most of the variance.

Figure 4.1.1 and Figure 4.1.2 shows the daily stock returns and cumulative stock returns for the training set respectively.

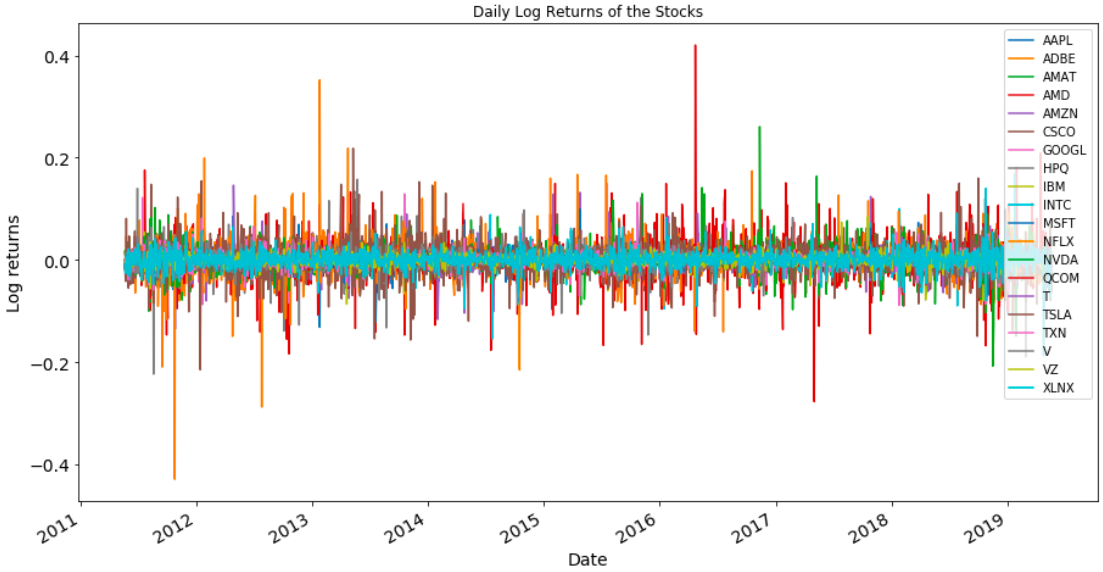

Figure 4.1.1: Daily stock returns

The raw data can be processed using Principal Component Analysis and the first principal component can be computed to reduce the overwhelming amount of data. Figure 4.1.3 shows the values for the first principal component.

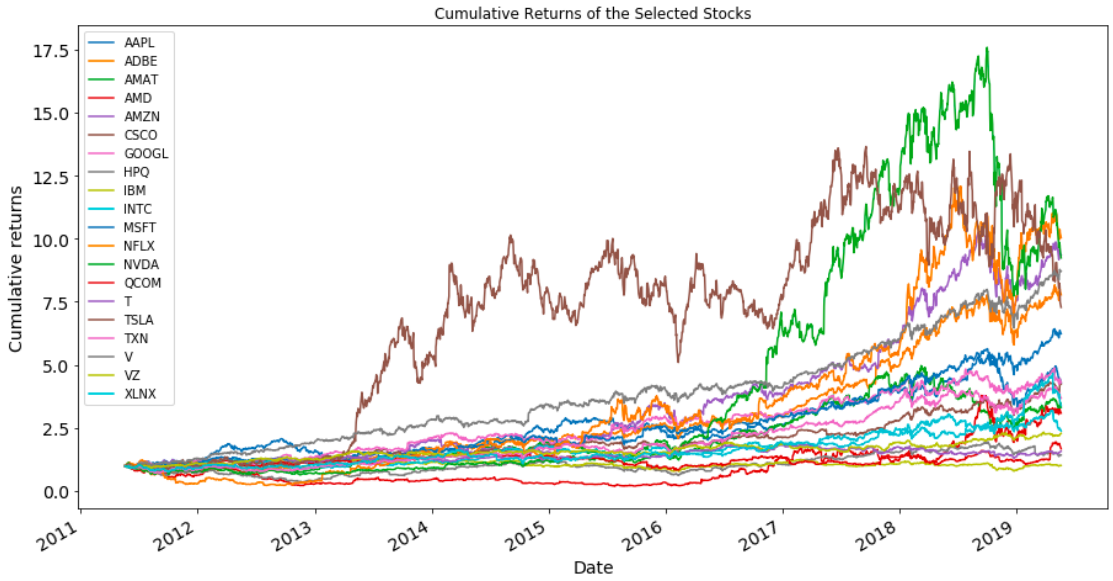

Figure 4.1.2: Cumulative returns

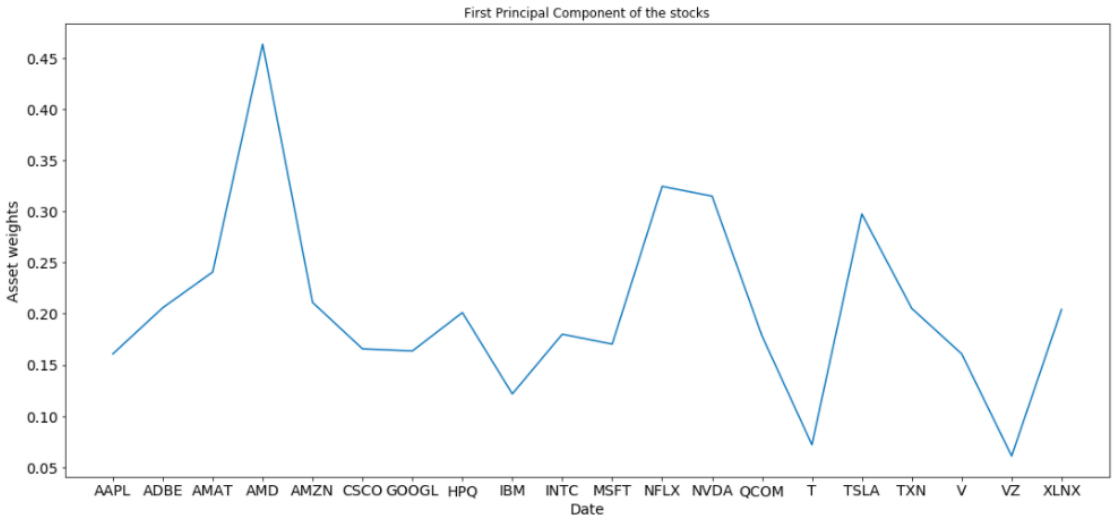

Figure 4.1.3: First principal component

The asset weights obtained from the first principal component can be used to formulate a portfolio of stocks by allocating the cash proportional to the weights. When the asset weights are applied on the testing set, it is observed that the PCA-weighted return replicates the non-weighted return. However, they are not exactly similar, as shown in Figure 4.1.4.

The performance of the PCA-weighted portfolio and the non-weighted portfolio can be evaluated using the the metrics discussed in section 3.3.5. Table 4.1.1 and 4.1.2 summarises the performance metrics for the non-weighted portfolio and PCA-weighted portfolio respectively.

Now using PCA, it is possible to select the best assets according to their first principal component weights. 10 stocks with the highest PCA weights were chosen to construct the portfolio. Figure 4.1.5 shows the difference in returns time series of top 10 stocks, PCA-weighted stocks, and unweighted stocks. Table 4.1.3 summarises the performance metrics for the top 10 stocks portfolio.

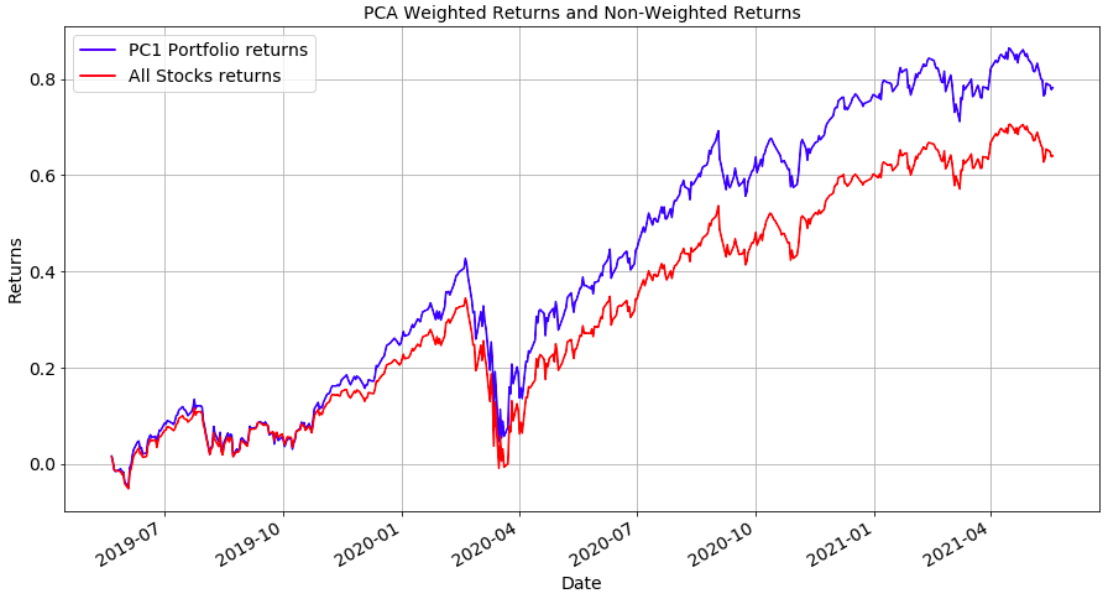

Figure 4.1.4: PCA-weighted portfolio returns and non-weighted stocks returns

Metric	Value
Annual return	31.35%
Cumulative returns	72.527%
Annual volatility	30.513%
Sharpe ratio	1.05
Maximum Drawdown	-32.197%
Skew	-0.96
Kurtosis	9.35

Table 4.1.1: Non-weighted portfolio performance

Metric	Value
Annual return	39.698%
Cumulative returns	95.156%
Annual volatility	33.344%
Sharpe ratio	1.17
Maximum Drawdown	-34.044%
Skew	-0.97
Kurtosis	7.87

Table 4.1.2: PCA-weighted portfolio performance

4.1.1 Performance Analysis

As observed in the tables 4.1.1-4.1.3, portfolio containing 10 stocks with the highest weights shows the best results in terms of annual return (46.763%), cumulative returns (115.393%), annual volatility (35.829%), kurtosis (6.00), and sharpe ratio (1.25). However, the PCA-weighted portfolio showed lower maximum drawdown (-34.044%),

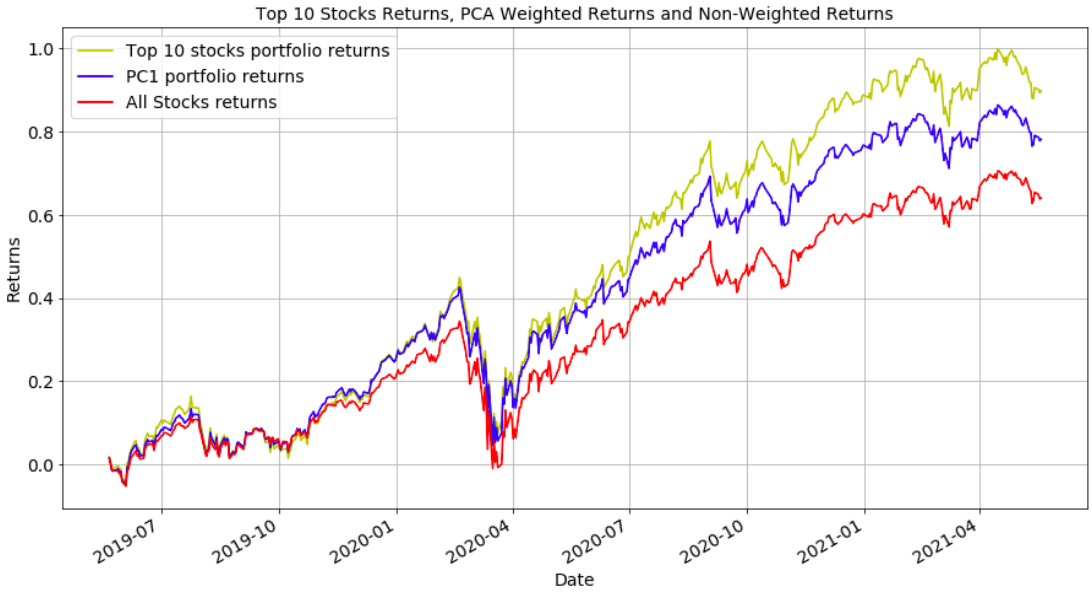

Figure 4.1.5: Top 10 stocks portfolio returns, PCA-weighted portfolio returns and non-weighted stocks returns

Metric	Value
Annual return	46.763%
Cumulative returns	115.393%
Annual volatility	35.829%
Sharpe ratio	1.25
Maximum Drawdown	-35.424%
Skew	-0.96
Kurtosis	6.00

Table 4.1.3: Top 10 PCA stocks portfolio performance

compared to the value for top 10 stocks portfolio (-35.424%). Moreover, all portfolios were equally skewed (around -0.96).

4.2 Independent Component Analysis

Independent Component Analysis (ICA) decomposes a given data into statistically independent components (ICs), however, the components are not arranged in the order of their variances. Similar to PCA, ICA can be explored to find dominant assets for portfolio construction. ICA can be applied to the daily stock returns data to find independent components, which are vectors containing asset weights just like principal components. However, unlike PCA, a method is required to identify the most significant independent component. Dominant independent components are defined as components with the largest maximum signal amplitude. They will have the largest effect on the weighted stock returns. [15]

"FastICA" algorithm is applied to the stocks returns data to find the independent components. 10 independent components were observed for the purpose of our analysis. Figure 4.2.1 shows the ICA-weighted returns series for the 10 independent components.

From Figure 4.2.1, it is deduced that the 5^{th} Independent Component has the maximum signal amplitude and

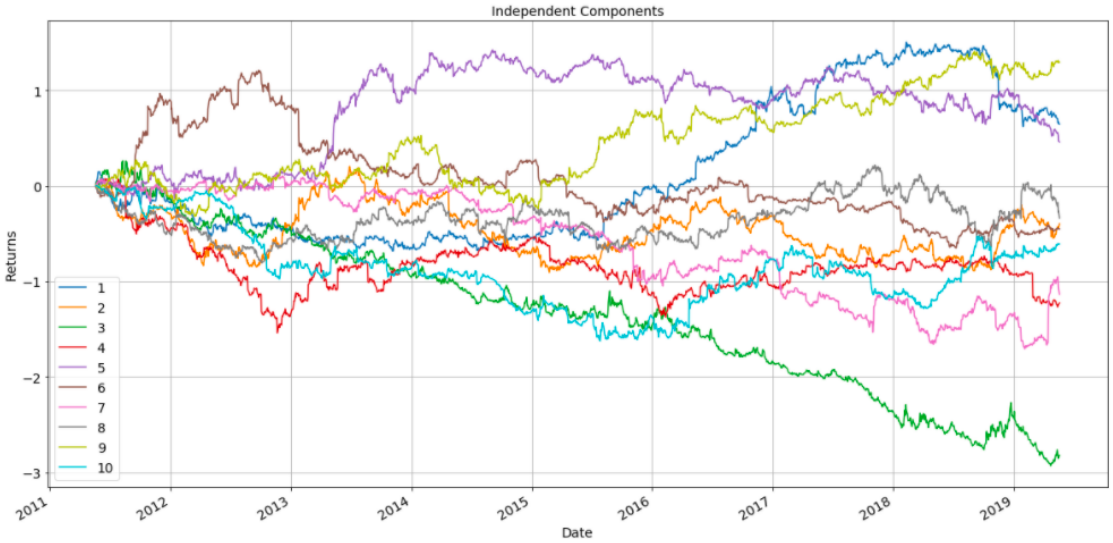

Figure 4.2.1: ICA-weighted returns series

demonstrates high returns for a longer duration compared to other independent components. Hence, it is selected as the dominant independent component. Figure 4.2.2 shows the asset weights for the dominant independent component.

The asset weights obtained from the dominant component can be used to formulate a portfolio by applying the weights vector obtained from training to the testing dataset. Similar shaped returns series are again observed in Figure 4.2.3. However, the ICA-weighted portfolio shows higher returns.

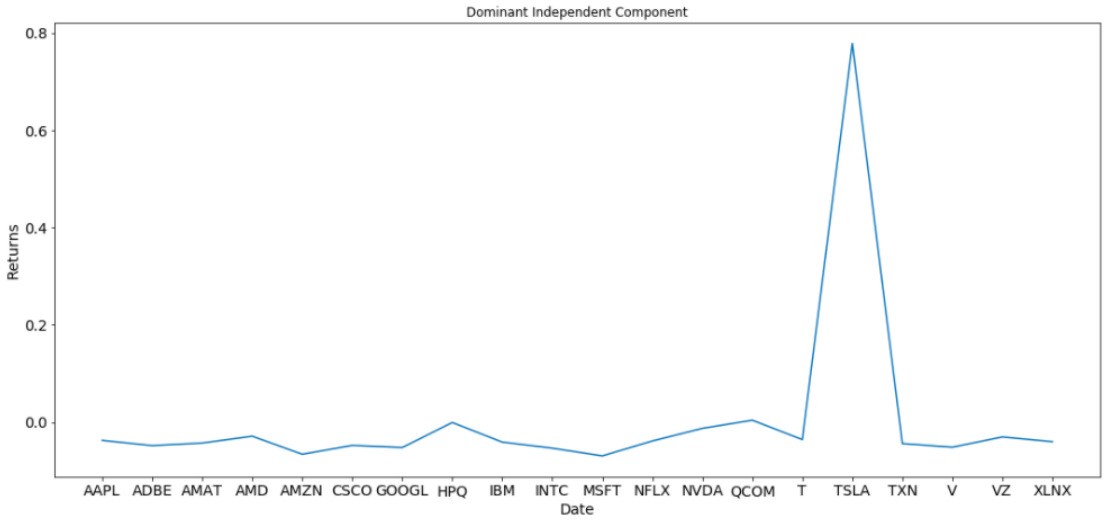

Figure 4.2.2: Dominant independent component

The performance of the ICA-weighted portfolio and the non-weighted portfolio can be evaluated using performance metrics, which are summarised in Table 4.2.1 and 4.2.2.

Just like PCA, it is possible to select 10 best assets according to their dominant independent component weights. Stocks with the highest ICA weights are selected to construct the portfolio. Figure 4.2.4 shows the difference in returns time series of top 10 stocks, ICA-weighted stocks, and unweighted stocks. Table 5.6 summarises the performance metrics for the top 10 stocks portfolio.

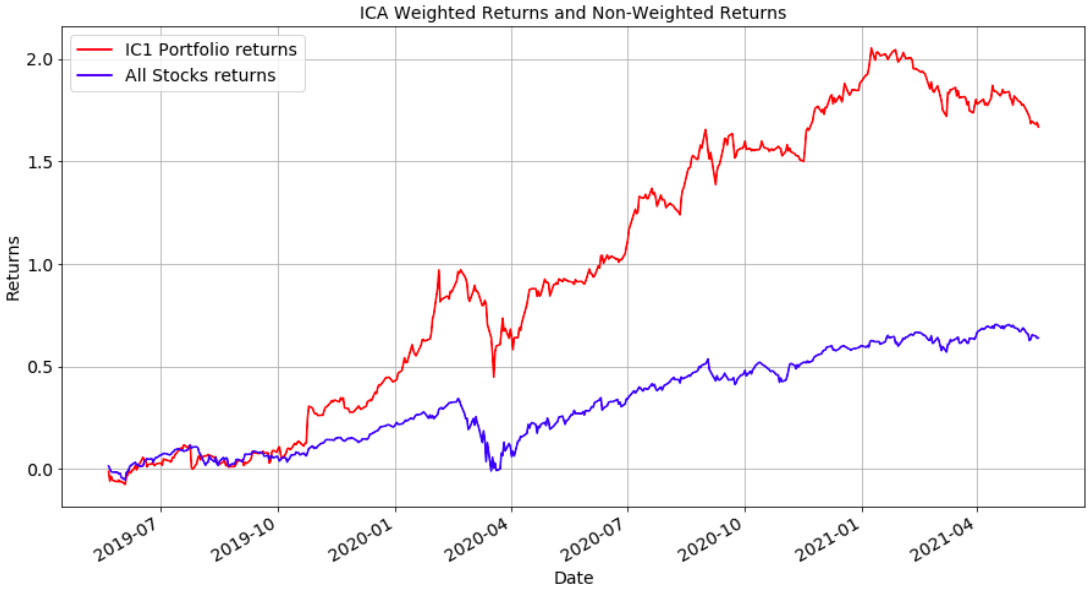

Figure 4.2.3: ICA-weighted portfolio returns and non-weighted stocks returns

Metric	Value
Annual return	31.35%
Cumulative returns	72.527%
Annual volatility	30.513%
Sharpe ratio	1.05
Maximum Drawdown	-32.197%
Skew	-0.96
Kurtosis	9.35

Table 4.2.1: Non-weighted portfolio performance

Metric	Value
Annual return	102.408%
Cumulative returns	309.691%
Annual volatility	50.589%
Sharpe ratio	1.65
Maximum Drawdown	-43.127%
Skew	-0.08
Kurtosis	3.96

Table 4.2.2: ICA-weighted portfolio performance

4.2.1 Performance Analysis

As observed in the tables 4.2.1-4.2.3, ICA-weighted portfolio showed the best results in terms of annual return (102.408%), cumulative returns (309.691%), annual volatility (50.589%), kurtosis (7.12), and sharpe ratio (1.65). The portfolio with 10 highest ICA weighted stocks showed lower maximum drawdown (-32.546%), which was closer

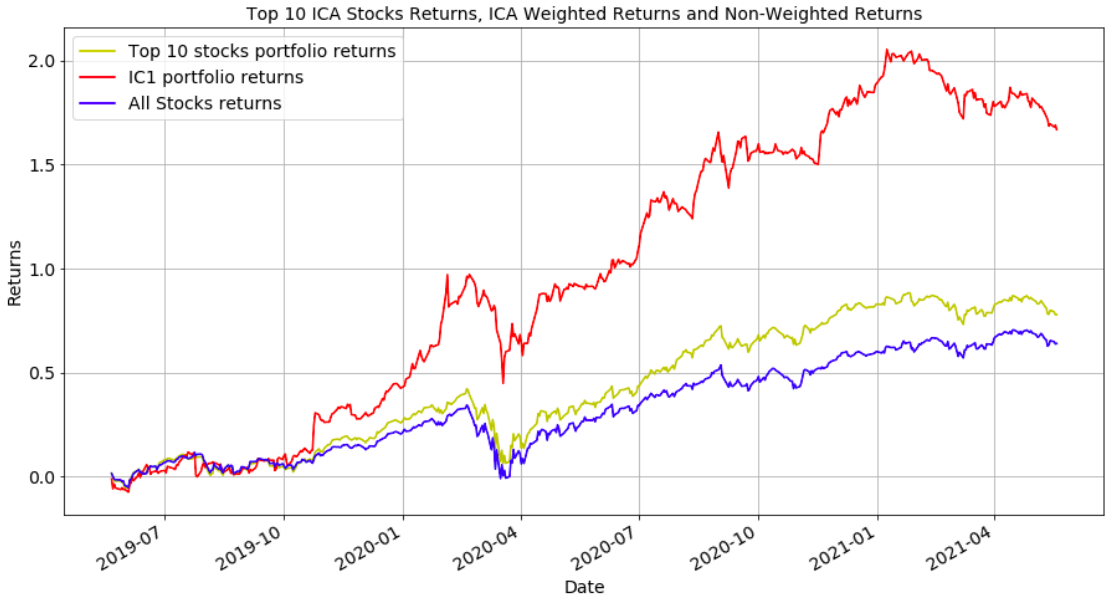

Figure 4.2.4: Top 10 stocks portfolio returns, ICA-weighted portfolio returns and non-weighted stocks returns

Metric	Value
Annual return	40.18%
Cumulative returns	96.505%
Annual volatility	31.891%
Sharpe ratio	1.22
Maximum Drawdown	-32.546%
Skew	-1.02
Kurtosis	7.12

Table 4.2.3: Top 10 ICA stocks portfolio performance

to the value for unweighted stocks (-32.197%). However, the distribution was highly skewed (-1.02). The skew value for all portfolios were around -0.96, except for the ICA-weighted portfolio (-0.08).

CHAPTER 5

Tensor Decomposition Methods for Portfolio Construction

To summarise this chapter, CPD and Tucker-based tensor decomposition algorithms such as CPD+ALS, HOSVD and HOOI are used to decompose the tensor into feature matrices. Statistical analysis methods such as PCA and ICA are applied on the assets feature matrix to find the dominant assets for the portfolio.

5.1 Data Tensorization

Tensor-based analysis involves tensorization of market data including adjusted closing prices and volume, and factors data including market capitalisation, 10-day momentum, price to book value and price to equity ratio and closing price predictive factor and closing price non-predictive factor to form a 3^{rd}-order tensor. Appendix B shows the first and last 5 rows of the stocks data. The three-mode stocks data tensor was formed consisting of time steps/dates, assets and stocks data. Since the tensor includes 2011 time steps from May 20,2011 to May 20, 2019 (training set), 20 assets and 8 market and factors data, the sizes of its modes are (2011, 20, 8). A depiction of the tensor and its decomposition can be observed in Figure 5.1.1. The tensor methods are not in any manner restricted to these number and sizes of modes, and more information regarding the stocks can be incorporated. More number of assets and factors data can be added to the tensor and the number of modes can be increased by adding more dimensions such as asset classes, sub-industries, etc. Moreover, other methods for tensorizing stocks data, such as a correlation-based approach has been explained in [74] and a three-way tensor representation has been explained in [75].

After data tensorization, tensor decomposition algorithms are applied on the stocks data tensor to decompose it into three feature matrices. Similar to the statistical analysis methods used in Chapter 4, statistical analysis is run on the assets feature matrix and portfolios are constructed by:

- Directly assigning computed asset weights to the stocks.
- Choosing top 10 dominant assets according to their asset weights.

5.2 Canonical Polyadic Decomposition

CPD is applied on the stock tensor to decompose it into its feature matrices. The kruskal rank (number of components) is randomly selected as 100. Since statistical analysis methods requires stationary data, it is important to make sure that the assets feature matrix is stationary. Figure 5.2.1 verifies that the data is stationary and PCA and ICA can be applied on the matrix.

Moreover, it is important to mention beforehand that the results obtained from CPD are inconsistent and it is

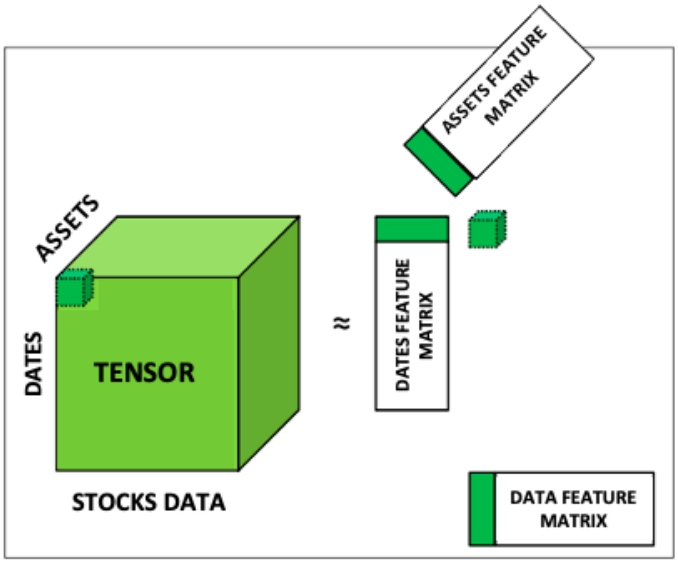

Figure 5.1.1: Tensor decomposition in 3 dimensions with the dates-assets-stock data setting

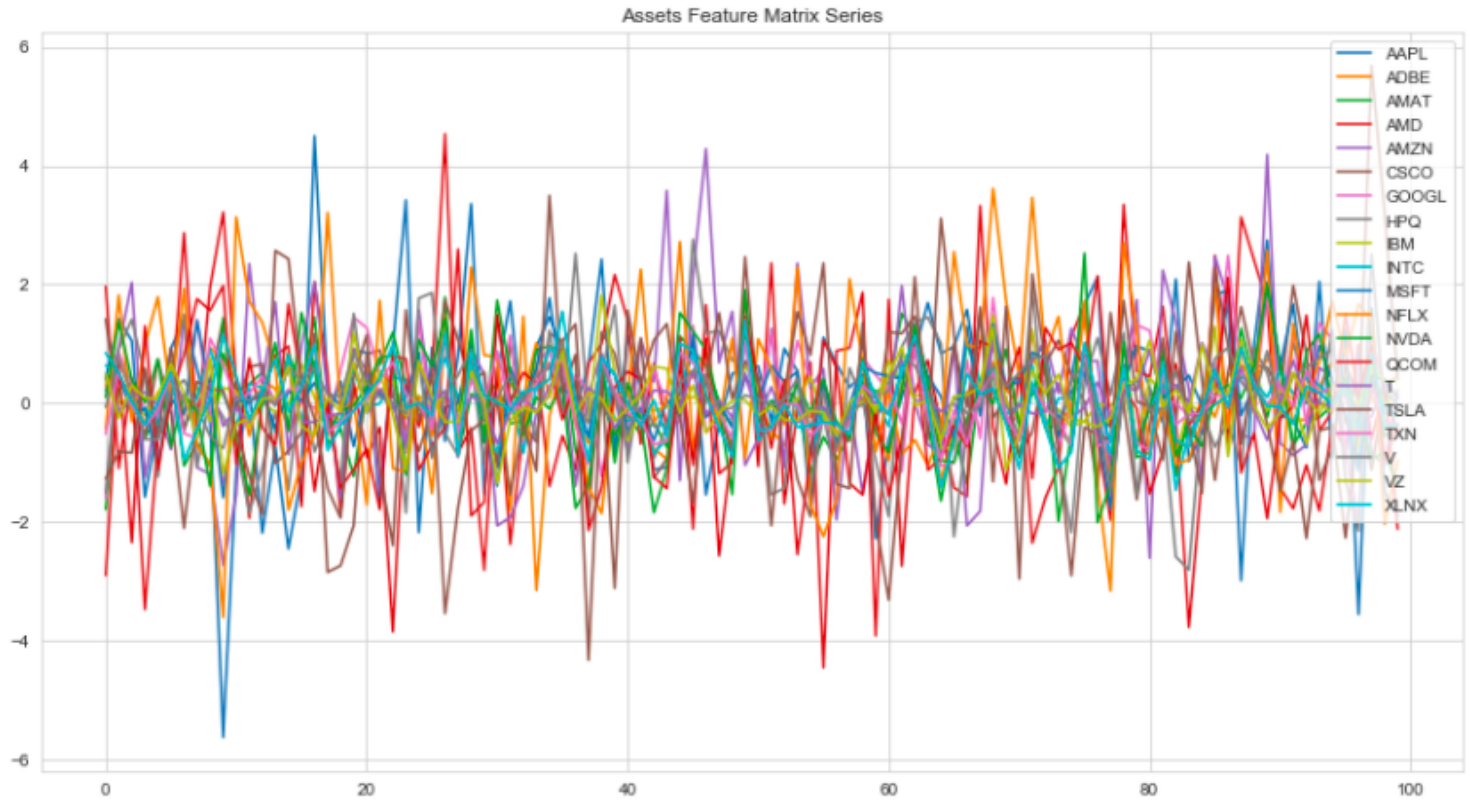

Figure 5.2.1: Assets feature matrix series for CPD

not possible to rely on performance results obtained from a single trial. Therefore, decomposition and analysis is run five times and the average performance metrics are computed. Performance metrics for all trials and the mean values are shown in the respective tables, however, only the figures for the second trial are displayed here to explain the implementation.

5.2.1 Principal Component Analysis

The assets feature matrix is processed using Principal Component Analysis and the asset weights for the first principal component are displayed in Figure 5.2.2.

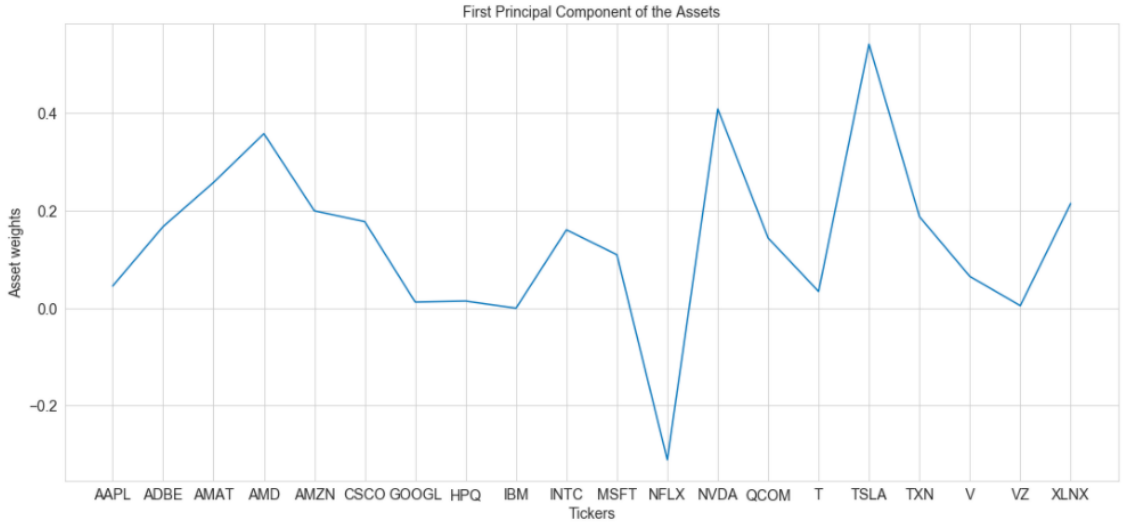

Figure 5.2.2: First Principal Component for CPD Analysis (Trial 2)

The asset weights obtained from the first principal component are used to formulate a portfolio of stocks. Figure 5.2.3 compares the reconstructed CPD+PCA-weighted returns and the unweighted returns. The performance metrics for CPD+PCA-weighted portfolio are summarised in Table 5.2.1.

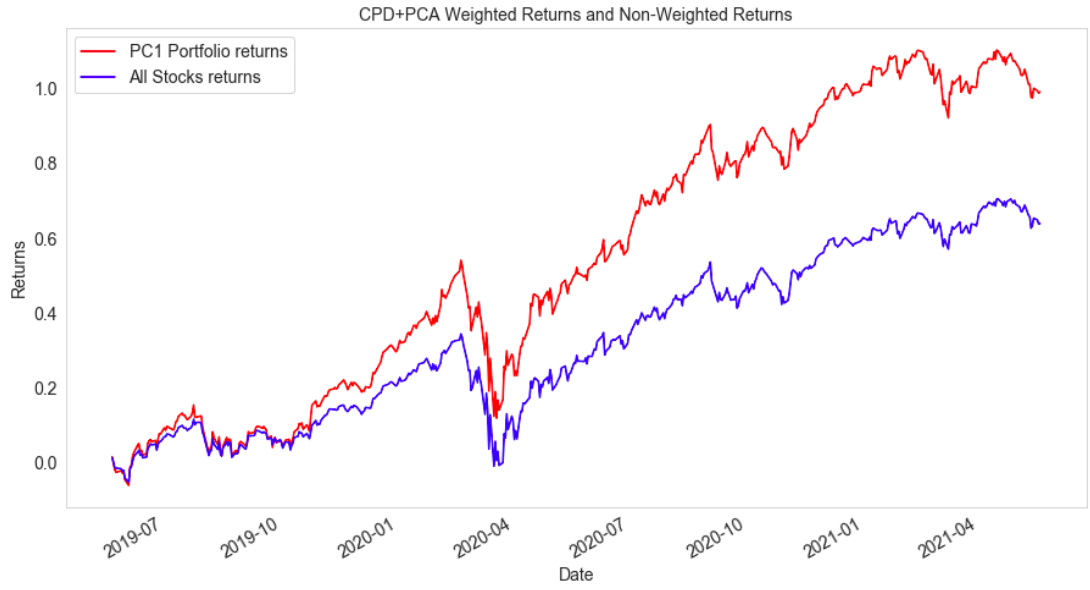

Figure 5.2.3: CPD+PCA-weighted portfolio returns and unweighted stocks returns (Trial 2)

Furthermore, 10 assets with the highest first principal component weights, as shown in Figure 5.2.4, are selected to construct a portfolio. Figure 5.2.5 compares the returns time series for the top 10 stocks portfolio with the PCA-weighted stocks and unweighted stocks. Table 5.2.2 summarises the performance metrics for the top 10 stocks portfolio.

Metric	Value 1	Value 2	Value 3	Value 4	Value 5	Mean
Annual return	41.393%	53.122%	63.001%	71.001%	49.925%	55.688%
Cumulative returns	99.92%	134.465%	165.692%	192.414%	124.774%	143.453%
Annual volatility	34.628%	36.994%	38.936%	39.573%	35.607%	37.148%
Sharpe ratio	1.18	1.34	1.45	1.56	1.32	1.37
Maximum Drawdown	-34.057%	-37.276%	-39.38%	-41.58%	-36.68%	-37.795%
Skew	-0.89	-0.96	-0.81	-0.99	-0.96	-0.92
Kurtosis	7.07	6.46	4.96	5.91	6.02	6.08

Table 5.2.1: CPD+PCA-weighted portfolio performance

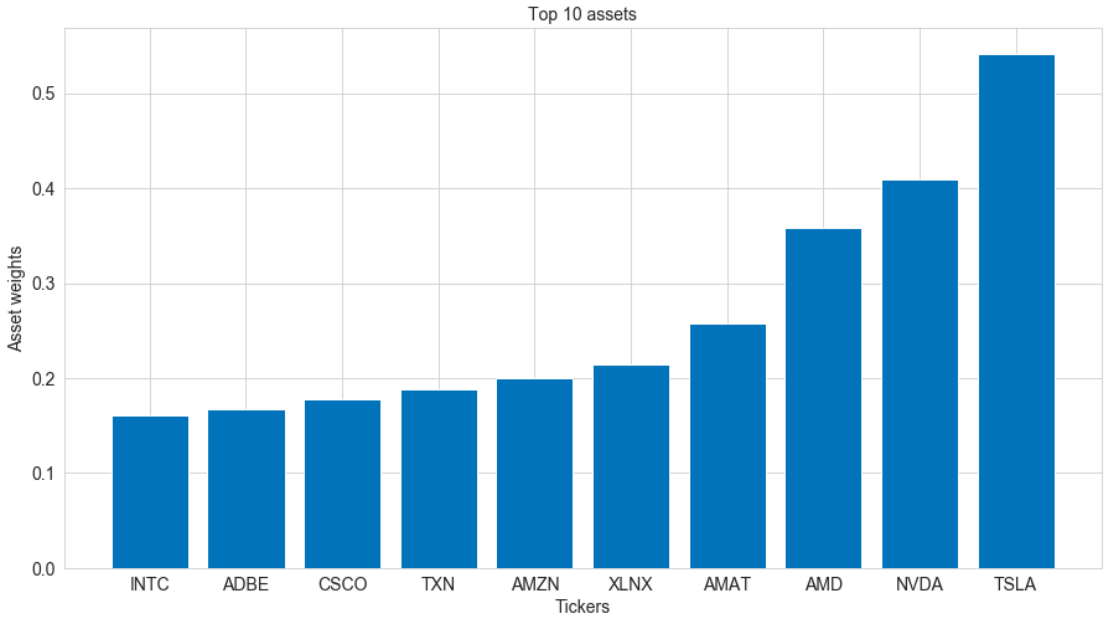

Figure 5.2.4: CPD+PCA top 10 stocks (Trial 2)

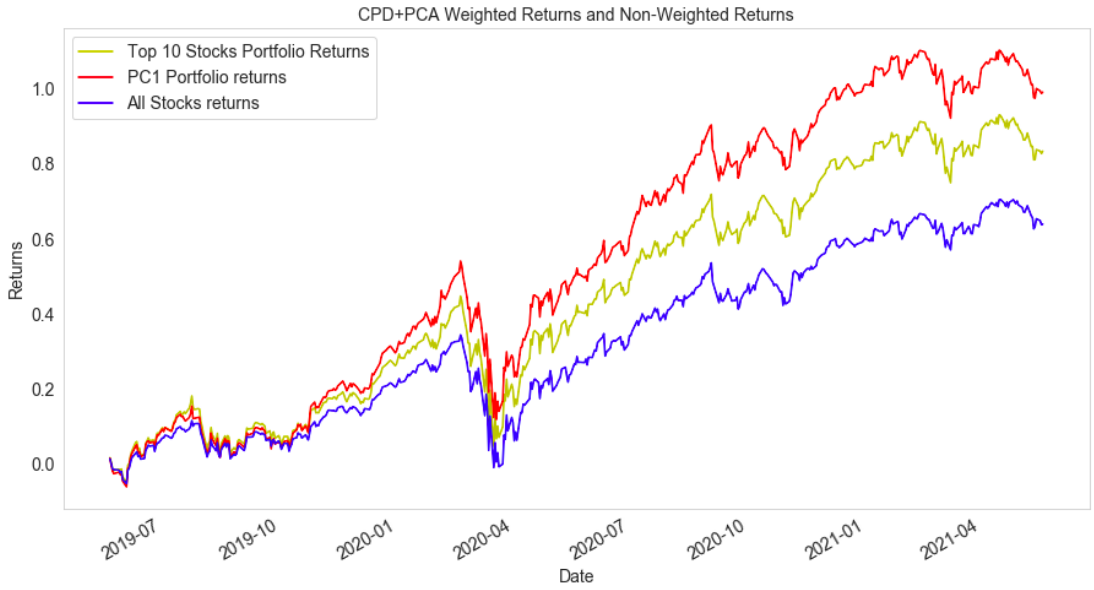

Figure 5.2.5: CPD+PCA Top 10 Stocks Portfolio Returns, CPD+PCA-weighted portfolio returns and unweighted stocks returns (Trial 2)

Metric	Value 1	Value 2	Value 3	Value 4	Value 5	Mean
Annual return	50.944%	42.041%	26.714%	18.008%	26.369%	32.815%
Cumulative returns	127.84%	101.755%	60.565%	39.258%	59.692%	77.822%
Annual volatility	34.529%	36.125%	31.372%	26.899%	29.249%	31.635%
Sharpe ratio	1.37	1.15	0.91	0.75	0.95	1.03
Maximum Drawdown	-35.358%	-35.58%	-31.457%	-29.765%	-33.482%	-33.128%
Skew	-1.10	-0.80	-0.67	-0.62	-0.88	-0.81
Kurtosis	7.69	6.67	7.33	11.28	8.63	8.32

Table 5.2.2: CPD+PCA top 10 stocks portfolio performance

5.2.2 Independent Component Analysis

The assets feature matrix are now processed using Independent Component Analysis and the dominant independent component is identified from the returns series for 10 Independent Components in Figure 5.2.6. It can be observed that the 3^{rd} Independent Component has the highest maximum signal amplitude. Although the maximum signal amplitude for the 7^{th} independent component is also one of the highest, the signal amplitude for the 3^{rd} component has been more consistently higher and hence, it is selected as the dominant independent component. The asset weights for the dominant independent component are shown in Figure 5.2.7.

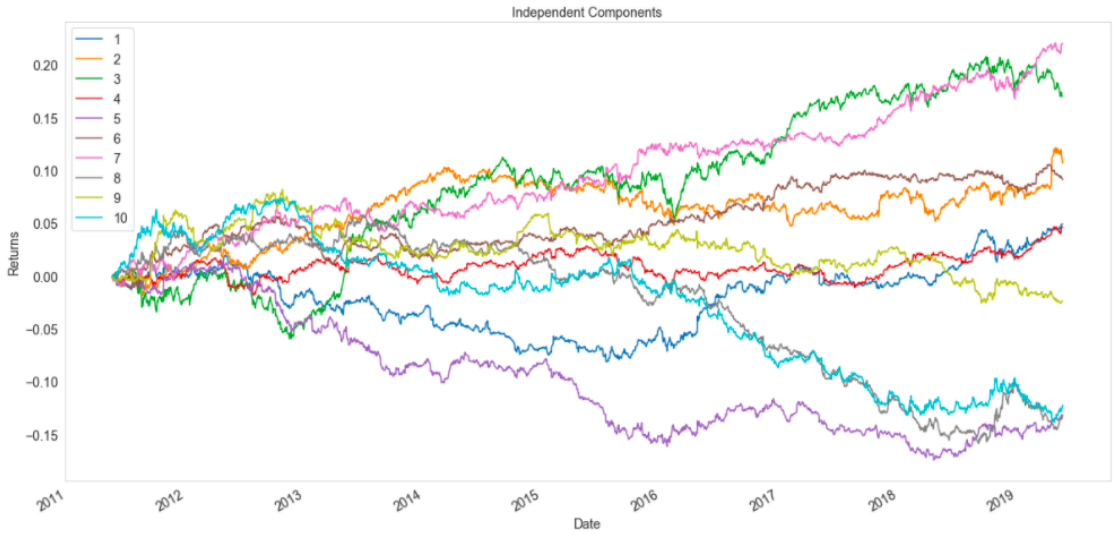

Figure 5.2.6: CPD+ICA-weighted returns series for all independent components (Trial 2)

The asset weights obtained from the dominant independent component are used to reconstruct the weighted portfolio. Figure 5.2.8 compares the reconstructed CPD+ICA-weighted returns and the unweighted returns. The performance metrics for CPD+ICA-weighted portfolio is summarised in Table 5.2.3.

Now 10 assets with the highest asset weights are selected, as shown in Figure 5.2.9. Figure 5.2.10 compares the returns time series for the CPD+ICA top 10 stocks, CPD+ICA-weighted stocks, and unweighted stocks together. Table 5.2.4 summarises the performance metrics for the CPD+ICA top 10 stocks portfolio.

5.2.3 Performance Analysis

As observed from the tables 5.2.1-5.2.4, the PCA-weighted portfolio performed the best in terms of annual return (55.688%), cumulative returns (143.453%), annual volatility (37.148 %), sharpe ratio (1.37) and skew (-0.92). The ICA-weighted portfolio performed the worse in terms of returns (6.371%) and volatility (4.647%), however it gave a favourable value for sharpe ratio (1.30), lowest kurtosis (5.136) and lowest maximum drawdown (-4.037%)

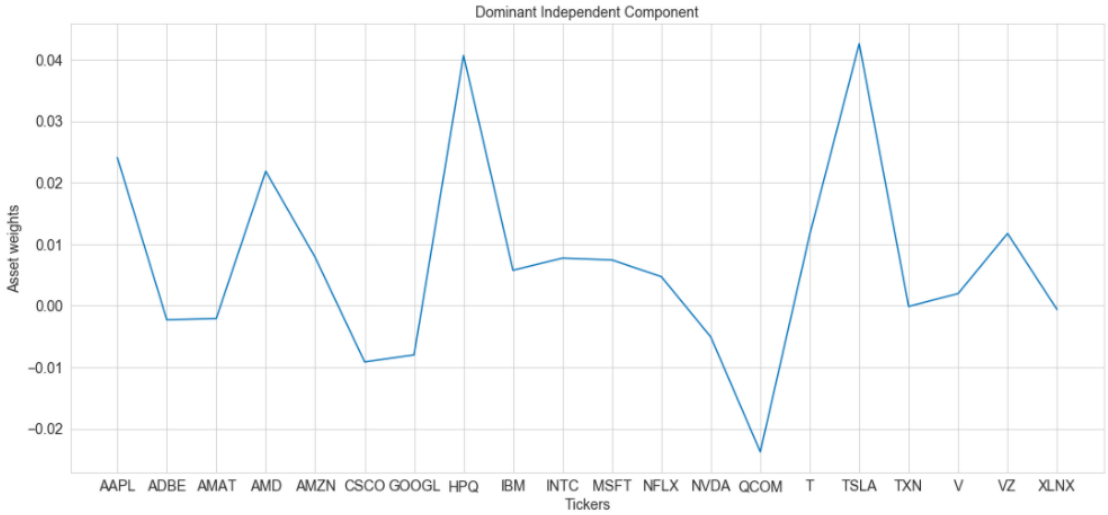

Figure 5.2.7: Dominant independent component for CPD analysis (Trial 2)

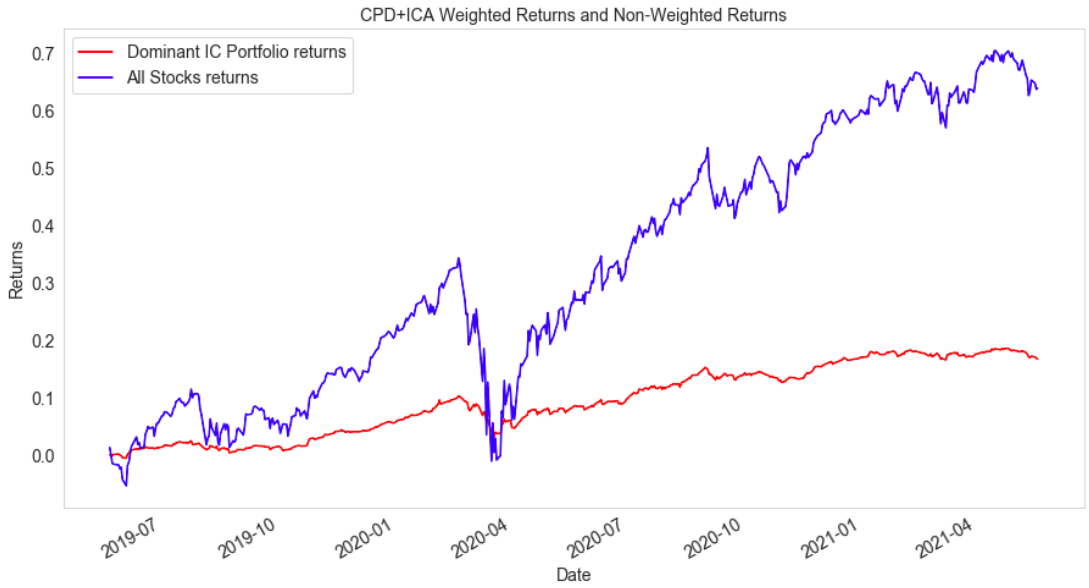

Figure 5.2.8: CPD+ICA-weighted portfolio returns and unweighted stocks returns (Trial 2)

Metric	Value 1	Value 2	Value 3	Value 4	Value 5	Mean
Annual return	7.917%	8.635%	2.136%	7.928%	5.241%	6.371%
Cumulative returns	16.461%	18.015%	4.319%	16.485%	10.757%	13.207%
Annual volatility	4.953%	5.183%	3.525%	4.718%	4.856%	4.647%
Sharpe ratio	1.56	1.62	0.62	1.64	1.08	1.30
Maximum Drawdown	-4.99%	-6.946%	-3.115%	-5.391%	-4.743%	-5.037%
Skew	-0.47	-1.06	-0.31	-0.82	-0.78	-0.69
Kurtosis	2.94	6.49	5.92	4.38	5.95	5.136

Table 5.2.3: CPD+ICA-weighted portfolio performance

among all portfolios. Moreover, inconsistencies in the decomposed factor matrices makes the results obtained from CPD-based analysis unreliable.

Figure 5.2.9: CPD+ICA top 10 stocks (Trial 2)

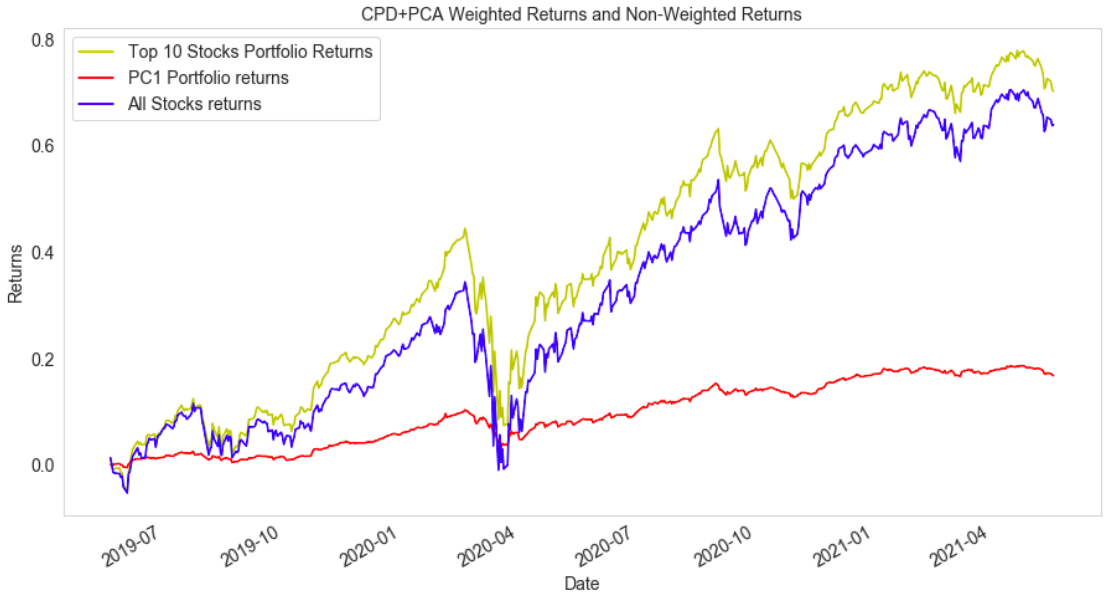

Figure 5.2.10: CPD+ICA top 10 stocks portfolio returns, CPD+ICA-weighted portfolio returns and unweighted stocks returns (Trial 2)

5.3 Tucker Decomposition: HOSVD Algorithm

Tucker decomposition is applied on the stock tensor to decompose the tensor into its feature matrices using the HOSVD algorithm and statistically analyse its assets feature matrix. The multi-linear rank for Tucker decomposition is selected as (2010,20,7), which means that number of components in the assets feature matrix is 20. The assets feature matrix series in Figure 5.3.1 verifies that the data is stationary and PCA and ICA can be applied on the matrix.

Metric	Value 1	Value 2	Value 3	Value 4	Value 5	Mean
Annual return	47.057%	36.047%	31.352%	45.822%	34.098%	38.875%
Cumulative returns	116.258%	85.089%	72.534%	112.639%	79.823%	93.269%
Annual volatility	36.401%	29.178%	31.58%	36.108%	33.75%	33.403%
Sharpe ratio	1.24	1.20	1.02	1.23	1.04	1.15
Maximum Drawdown	-34.674%	-33.233%	-31.719%	-36.528%	-33.138%	-33.858%
Skew	-0.81	-1.19	-0.96	-0.98	-0.67	-0.92
Kurtosis	6.12	9.82	9.65	6.95	6.42	7.79

Table 5.2.4: CPD+ICA top 10 stocks portfolio performance

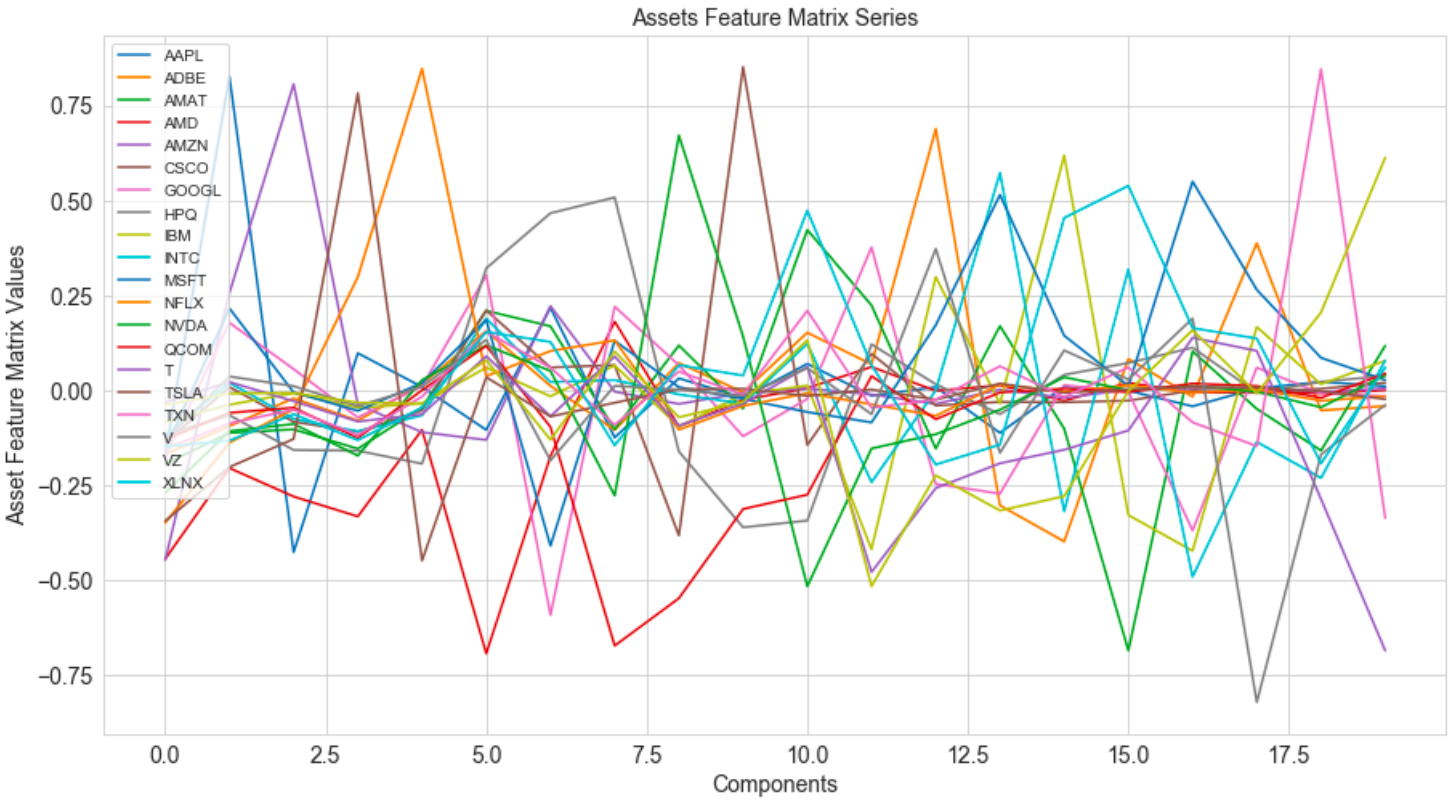

Figure 5.3.1: Assets feature matrix series for HOSVD algorithm

5.3.1 Principal Component Analysis

The assets feature matrix is processed using Principal Component Analysis and the first principal component weights are computed. Figure 5.3.2 shows the asset weights for the first principal component.

The asset weights obtained from the first principal component are used to reconstruct the HOSVD+PCA-weighted portfolio returns, which is shown in Figure 5.3.3. The performance metrics for HOSVD+PCA-weighted portfolio is summarised in Table 5.3.1.

To construct the portfolio with best assets, 10 assets with the highest first principal component values are selected. The selected stocks are displayed in Figure 5.3.4 and the returns time series for HOSVD+PCA top 10 stocks is compared to the HOSVD+PCA-weighted returns series, and unweighted stocks returns series in Figure 5.3.5. Table 5.3.2 summarises the performance metrics for HOSVD+PCA top 10 stocks portfolio.

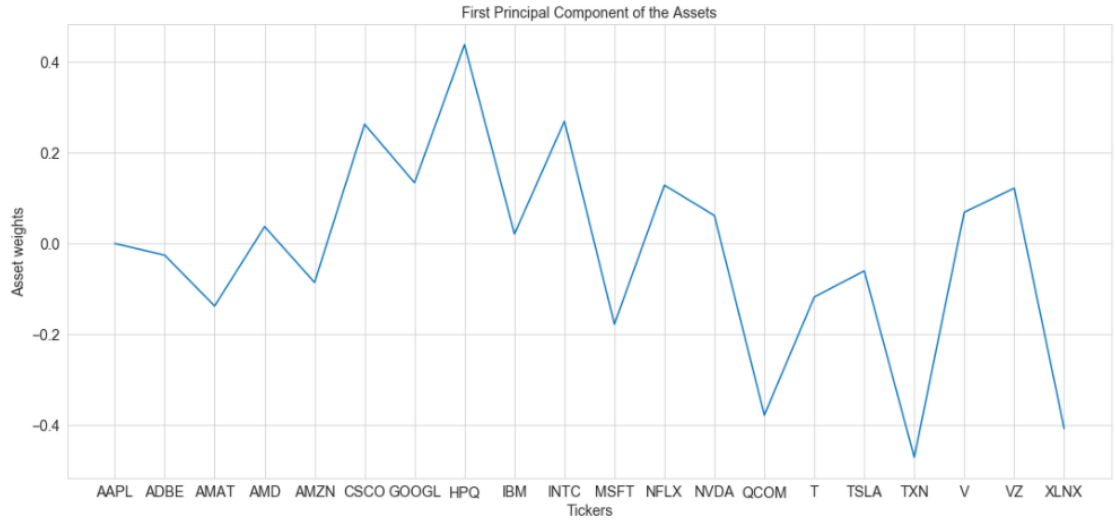

Figure 5.3.2: First principal component for HOSVD-based analysis

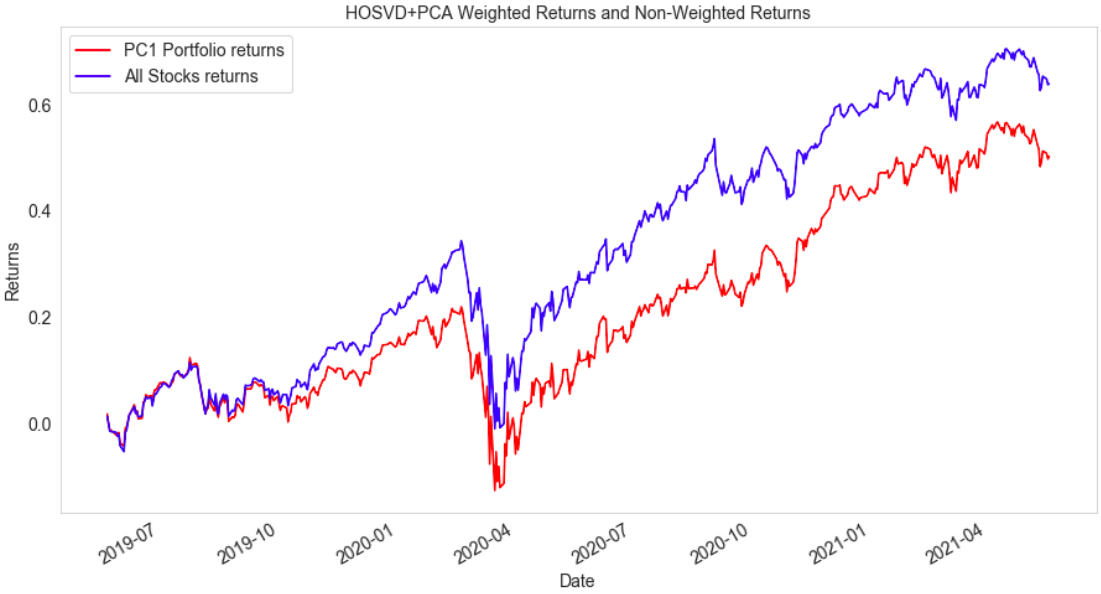

Figure 5.3.3: HOSVD+PCA-weighted portfolio returns and unweighted stocks returns

Metric	Value
Annual return	22.314%
Cumulative returns	49.607%
Annual volatility	31.468%
Sharpe ratio	0.80
Maximum Drawdown	-31.359%
Skew	-0.75
Kurtosis	8.10

Table 5.3.1: HOSVD+PCA-weighted portfolio performance

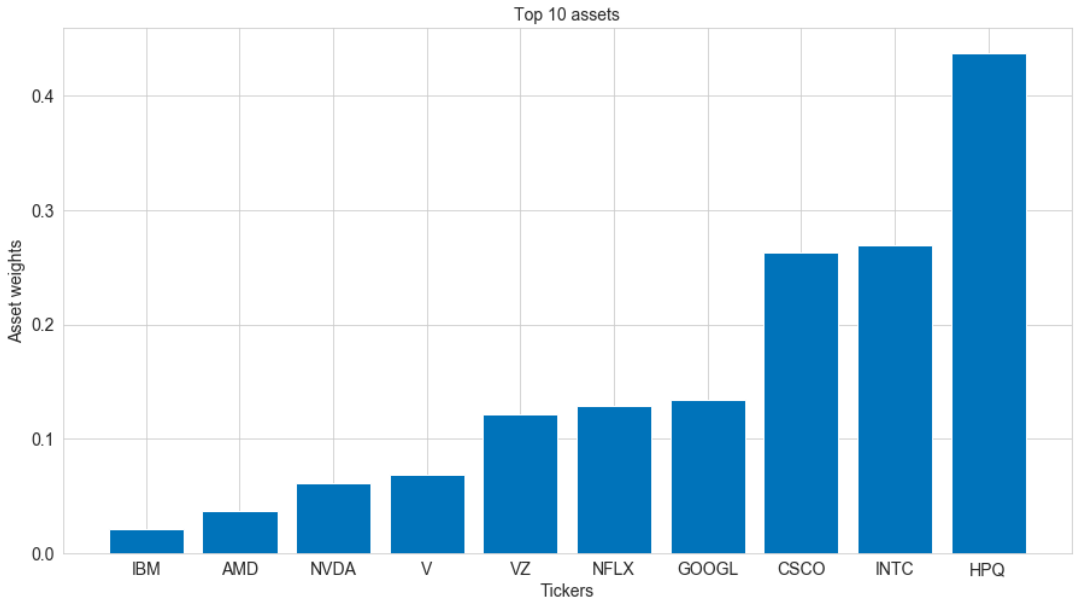

Figure 5.3.4: HOSVD+PCA top 10 stocks

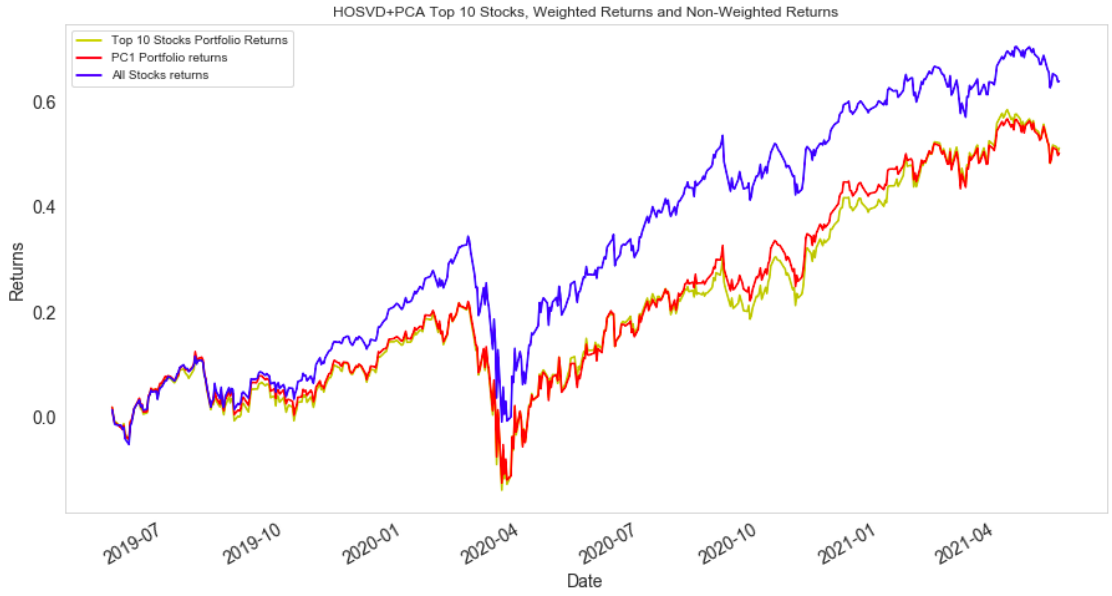

Figure 5.3.5: HOSVD+PCA top 10 stocks portfolio returns, HOSVD+PCA-weighted portfolio returns and unweighted stocks returns

5.3.2 Independent Component Analysis

The assets feature matrix is analysed using ICA and the dominant independent component is selected by observing the returns series for all 10 Independent Components in Figure 5.3.6. It is visible that the 8^{th} independent component has the highest maximum signal amplitude and is selected as the dominant independent component. The dominant independent component asset weights are displayed in Figure 5.3.7.

The reconstructed HOSVD+ICA-weighted returns and the unweighted returns are shown in Figure 5.3.8. The performance metrics for HOSVD+ICA-weighted portfolio is also summarised in Table 5.3.3.

10 assets with the highest asset weights, as shown in Figure 5.3.9, are selected to construct the portfolio and the returns time series for HOSVD+ICA top 10 stocks, HOSVD+ICA-weighted stocks, and unweighted stocks are compared in Figure 5.3.10. Table 5.3.4 summarises the performance metrics for HOSVD+ICA top 10 stocks

Metric	Value
Annual return	22.491%
Cumulative returns	50.039%
Annual volatility	32.462%
Sharpe ratio	0.79
Maximum Drawdown	-32.383%
Skew	-0.78
Kurtosis	8.89

Table 5.3.2: HOSVD+PCA top 10 stocks portfolio performance

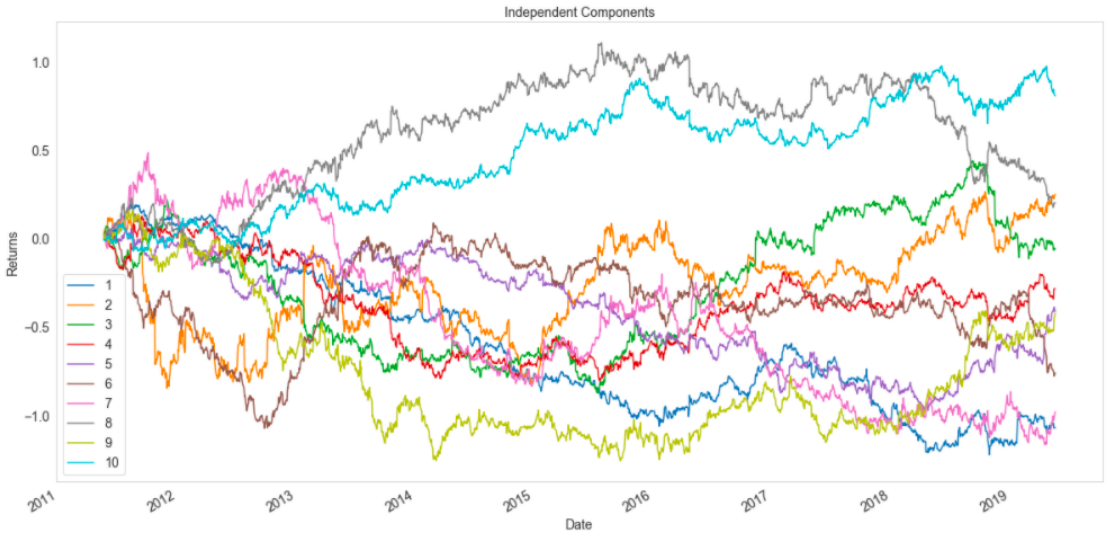

Figure 5.3.6: HOSVD+ICA-weighted returns series for all independent components

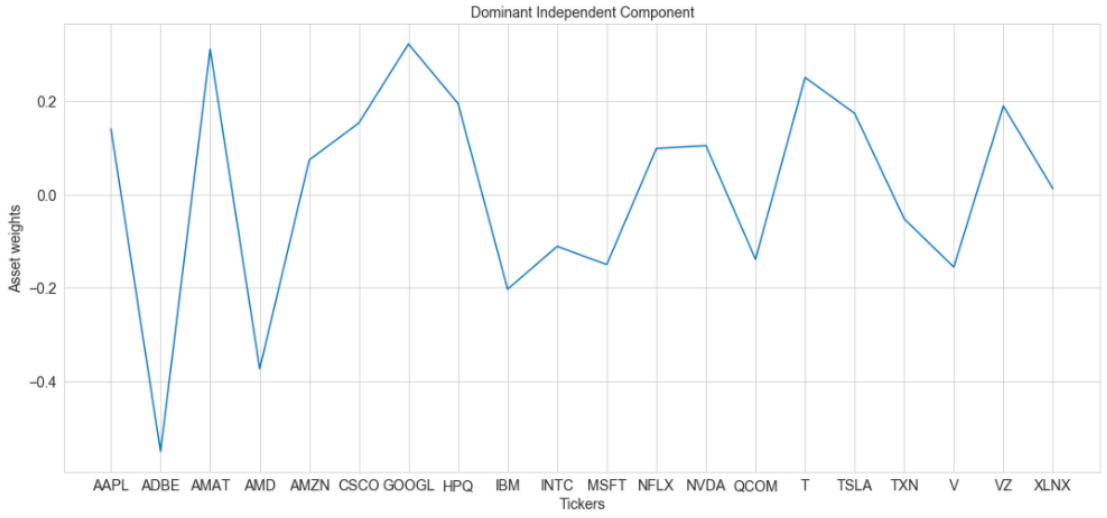

Figure 5.3.7: Dominant independent component for HOSVD analysis

portfolio.

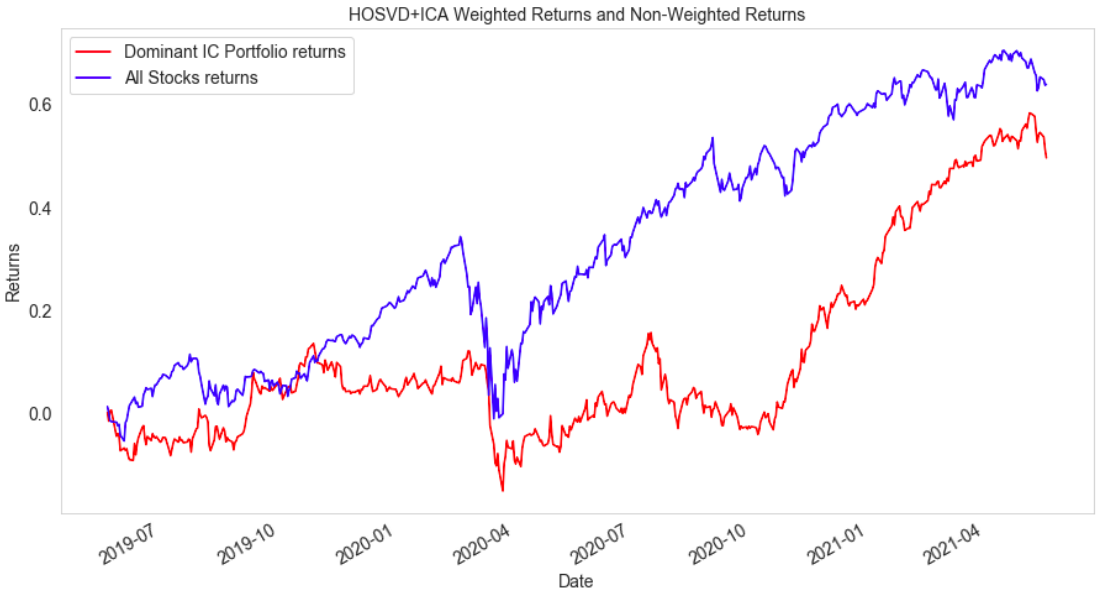

Figure 5.3.8: HOSVD+ICA-weighted portfolio returns and unweighted stocks returns

Metric	Value
Annual return	24.222%
Cumulative returns	54.312%
Annual volatility	25.086%
Sharpe ratio	0.99
Maximum Drawdown	-25.79%
Skew	-0.33
Kurtosis	1.20

Table 5.3.3: HOSVD+ICA-weighted portfolio performance

Metric	Value
Annual return	40.845%
Cumulative returns	98.373%
Annual volatility	30.359%
Sharpe ratio	1.28
Maximum Drawdown	-33.726%
Skew	-1.28
Kurtosis	10.29

Table 5.3.4: HOSVD+ICA top 10 stocks portfolio performance

5.3.3 Performance Analysis

As observed from the tables 5.3.1-5.3.4, portfolio constructed using the top 10 stocks selected by applying ICA on the assets feature matrix gives the most favourable results in terms of annual return (40.845%), cumulative returns (98.373%), annual volatility (30.359%), sharpe ratio (1.28) and skew (-1.28). The ICA-weighted portfolio

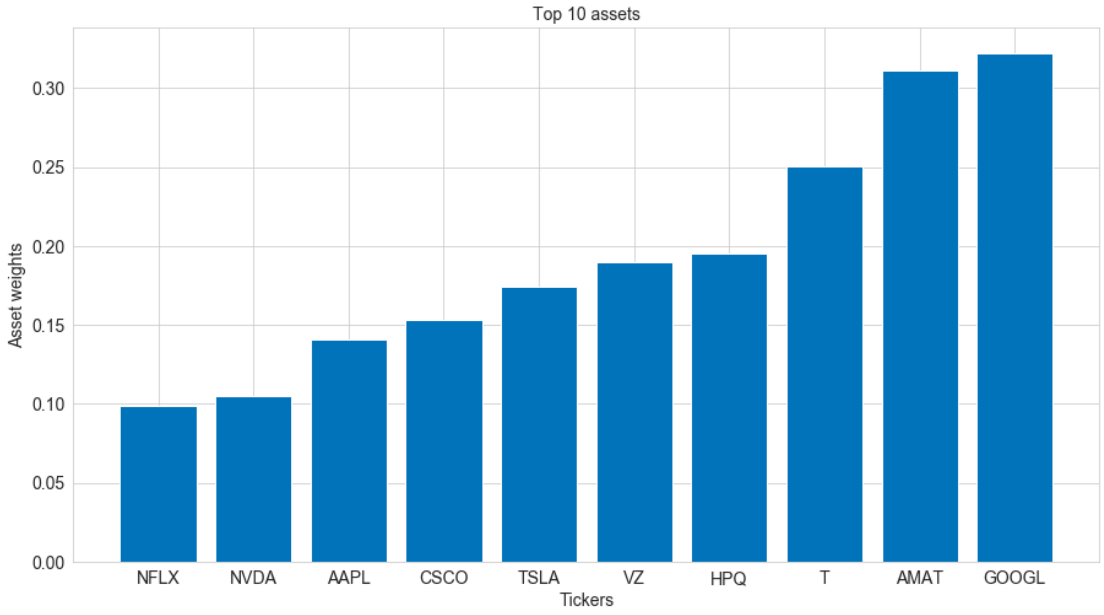

Figure 5.3.9: HOSVD+ICA top 10 stocks

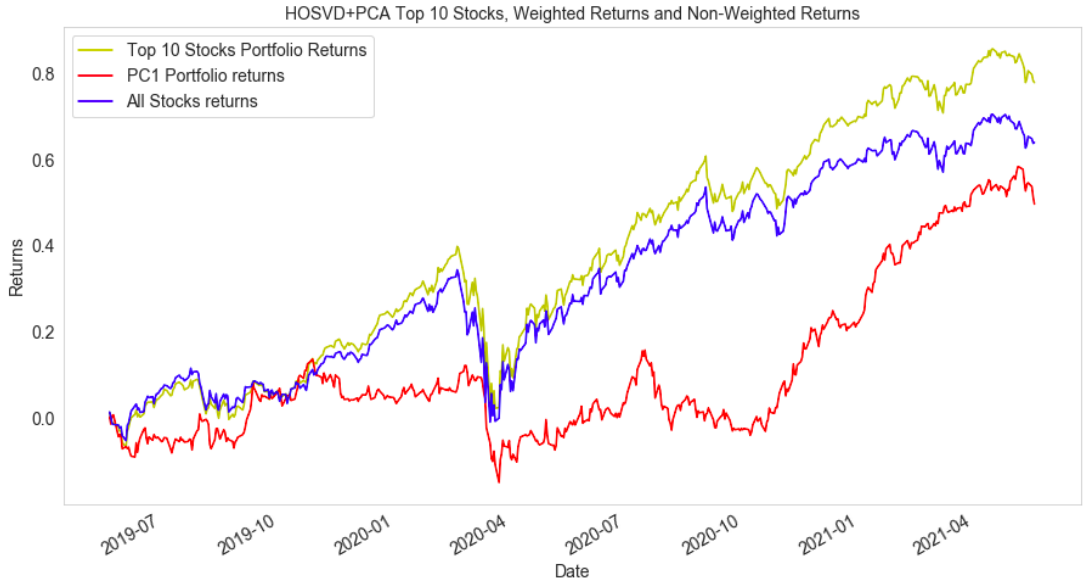

Figure 5.3.10: HOSVD+ICA top 10 stocks portfolio returns, HOSVD+ICA-weighted portfolio returns and unweighted stocks returns

performed better than both PCA-weighted and PCA top 10 stocks portfolios in terms of returns, however, it showed the least favourable value for annual volatility (25.086%). Most favourable values for kurtosis (1.20) and maximum drawdown (-25.79%) were obtained for the ICA-weighted portfolio. Moreover, portfolios based on HOSVD algorithm have been more persistent compared to CPD-based portfolios.

5.4 Tucker Decomposition: HOOI Algorithm

HOOI algorithm is applied for Tucker decomposition of the stocks tensor with the same multi-linear rank as used for HOSVD algorithm, i.e., (2010,20,7). Figure 5.4.1 verifies the stationarity of assets feature matrix for HOOI algorithm.

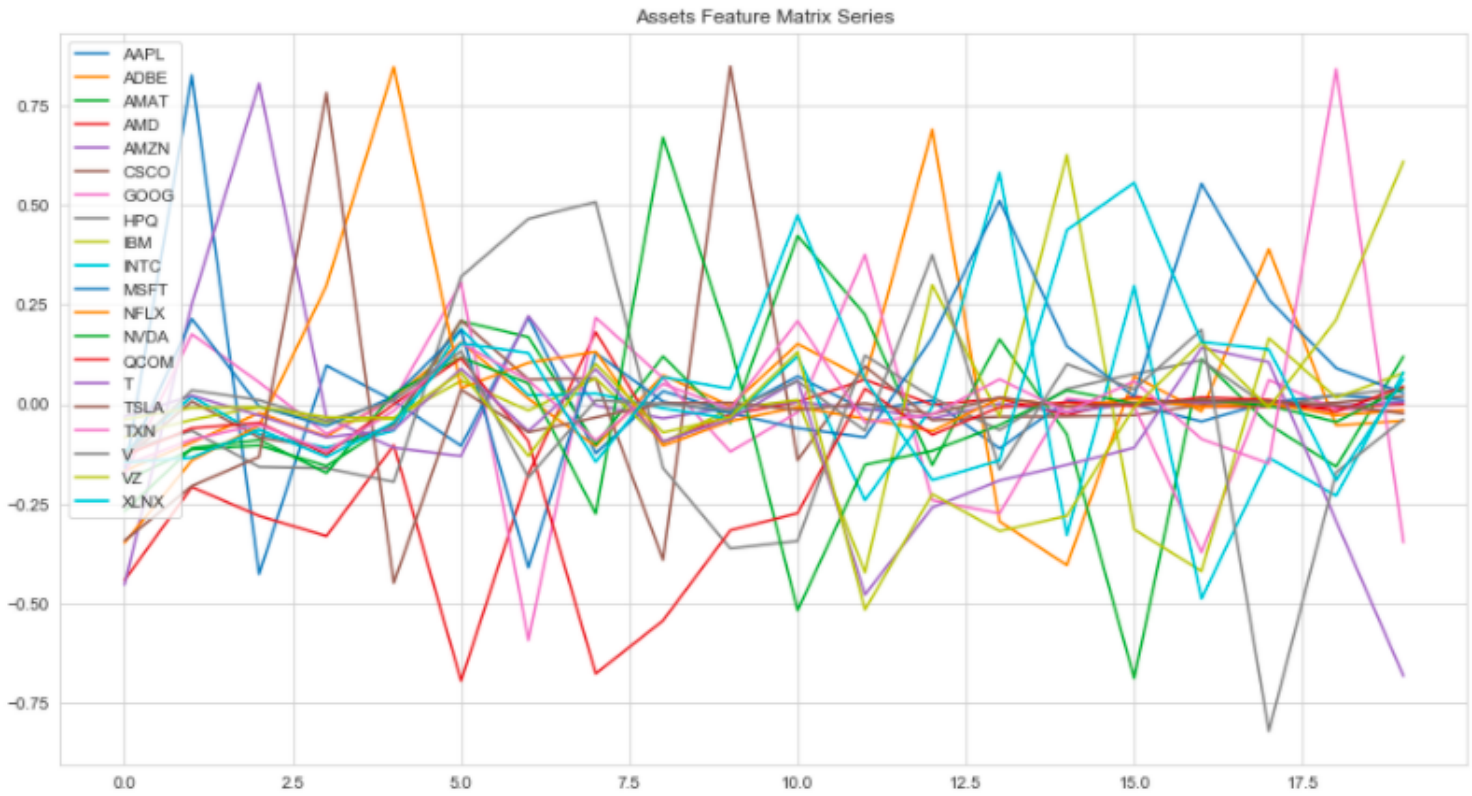

Figure 5.4.1: Assets feature matrix Series for HOOI algorithm

5.4.1 Principal component analysis

The first principal component asset weights are computed and displayed in Figure 5.4.2.

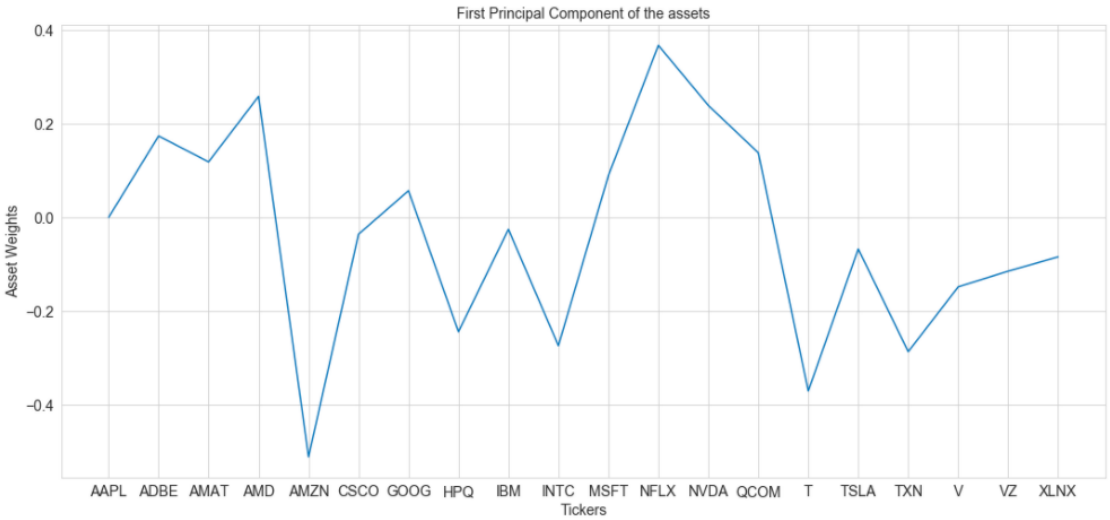

Figure 5.4.2: First principal component for HOOI-based analysis

These asset weights are applied on respective assets in the returns data in the testing set to reconstruct HOOI+PCA-weighted returns and compare it to the unweighted returns in Figure 5.4.3. The performance metrics for HOOI+PCA-weighted portfolio is summarised in Table 5.4.1.

Furthermore, 10 assets with the highest first principal component weights are selected to construct the portfolio, as

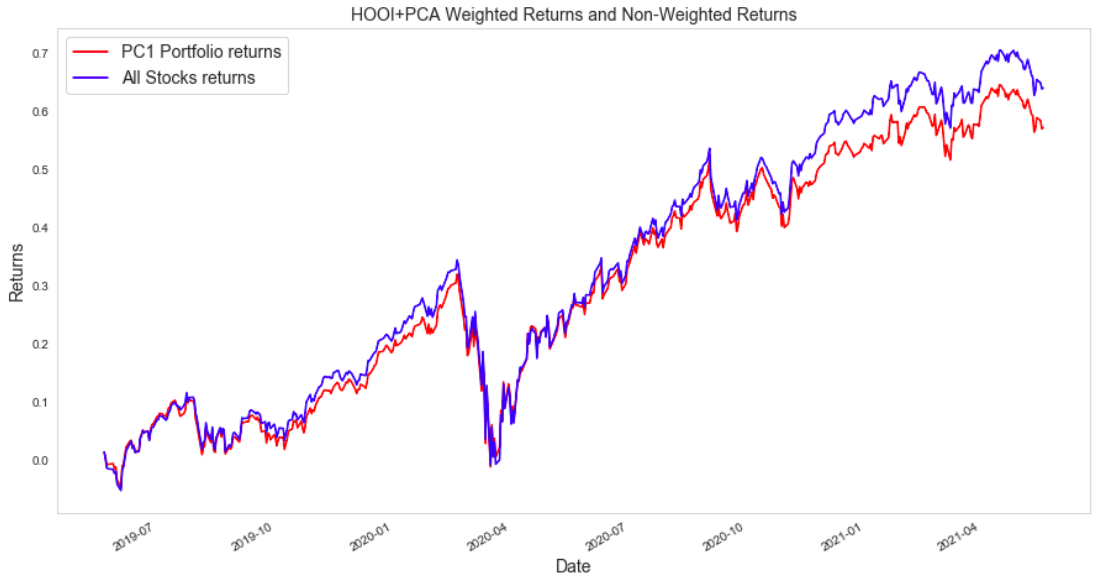

Figure 5.4.3: HOOI+PCA-weighted portfolio returns and unweighted stocks returns

Metric	Value
Annual return	27.313%
Cumulative returns	62.085%
Annual volatility	29.769%
Sharpe ratio	0.96
Maximum Drawdown	-30.361%
Skew	-0.88
Kurtosis	8.69

Table 5.4.1: HOOI+PCA-weighted portfolio performance

shown in Figure 5.4.4. Figure 5.4.5 compares the returns time series for HOOI+PCA top 10 stocks, HOOI+PCA-weighted stocks, and unweighted stocks together and the performance metrics for HOOI+PCA top 10 stocks portfolio are summarised in Table 5.4.2.

Metric	Value
Annual return	37.719%
Cumulative returns	89.664%
Annual volatility	33.879%
Sharpe ratio	1.12
Maximum Drawdown	-33.678%
Skew	-0.90
Kurtosis	8.76

Table 5.4.2: HOOI+PCA top 10 stocks portfolio performance

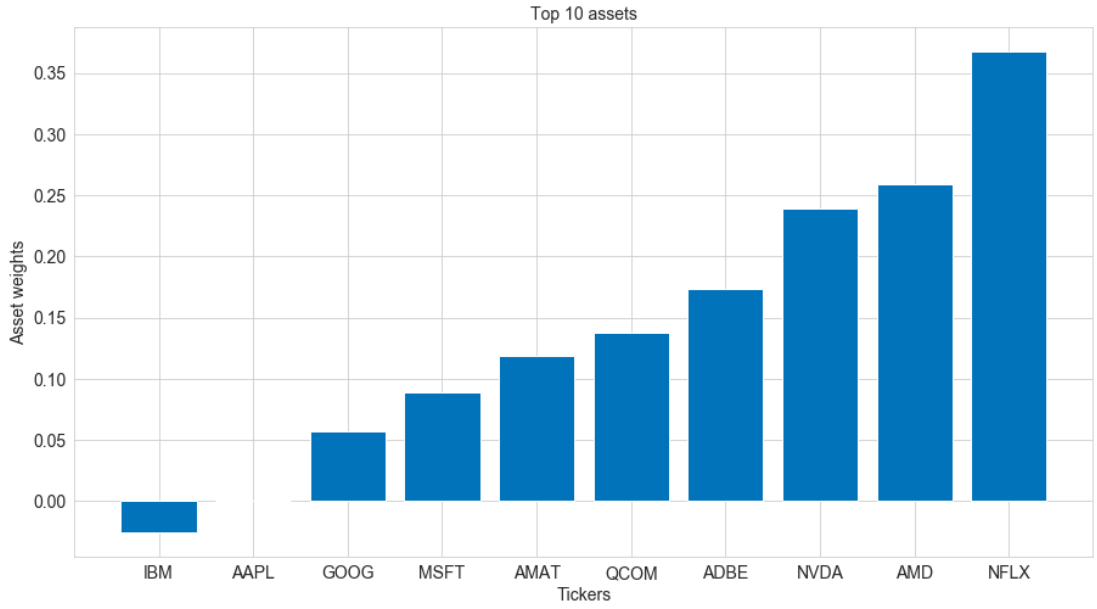

Figure 5.4.4: HOOI+PCA top 10 stocks

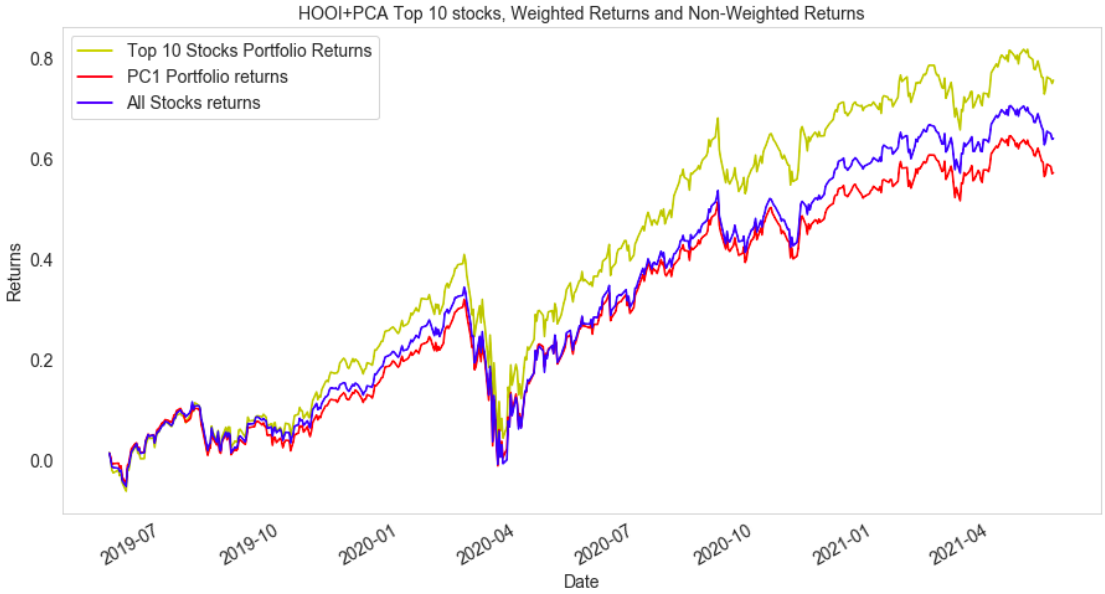

Figure 5.4.5: HOOI+PCA top 10 stocks portfolio returns, HOOI+PCA-weighted portfolio returns and unweighted stocks returns

5.4.2 Independent Component Analysis

Now the assets feature matrix is processed using Independent Component Analysis and the dominant independent component is selected from the weighted returns time series for the 10 independent components in Figure 5.4.6. The 7^{th} independent component has the highest maximum signal amplitude. The asset weights for the dominant independent component are shown in Figure 5.4.7.

The asset weights obtained from the dominant independent component are used to reconstructed the HOOI+ICA-weighted returns and the unweighted returns and compare them in Figure 5.4.8. The performance metrics for HOOI+ICA-weighted portfolio is summarised in Table 5.4.3.

10 assets with the highest weights are selected, as shown in Figure 5.4.9. Figure 5.4.10 compares the returns time

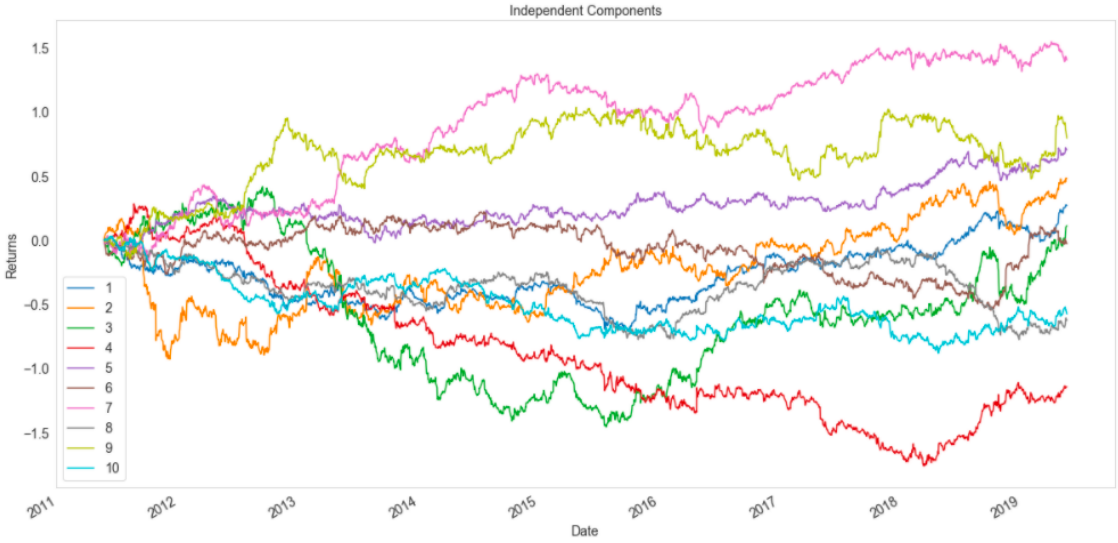

Figure 5.4.6: HOOI+ICA-weighted returns series for all independent components

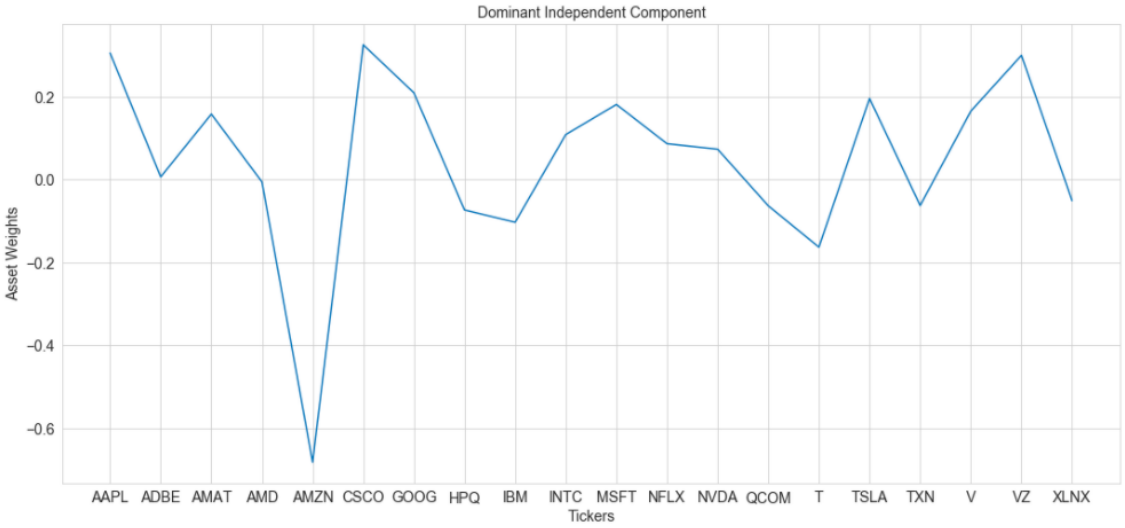

Figure 5.4.7: Dominant independent component for HOOI-based analysis

Metric	Value
Annual return	51.331%
Cumulative returns	129.012%
Annual volatility	38.207%
Sharpe ratio	1.28
Maximum Drawdown	-41.088%
Skew	-1.21
Kurtosis	13.81

Table 5.4.3: HOOI+ICA-weighted portfolio performance

series for HOOI+ICA top 10 stocks, HOOI+ICA-weighted stocks, and unweighted stocks and the performance metrics for HOOI+ICA top 10 stocks portfolio are summarised in Table 5.4.4.

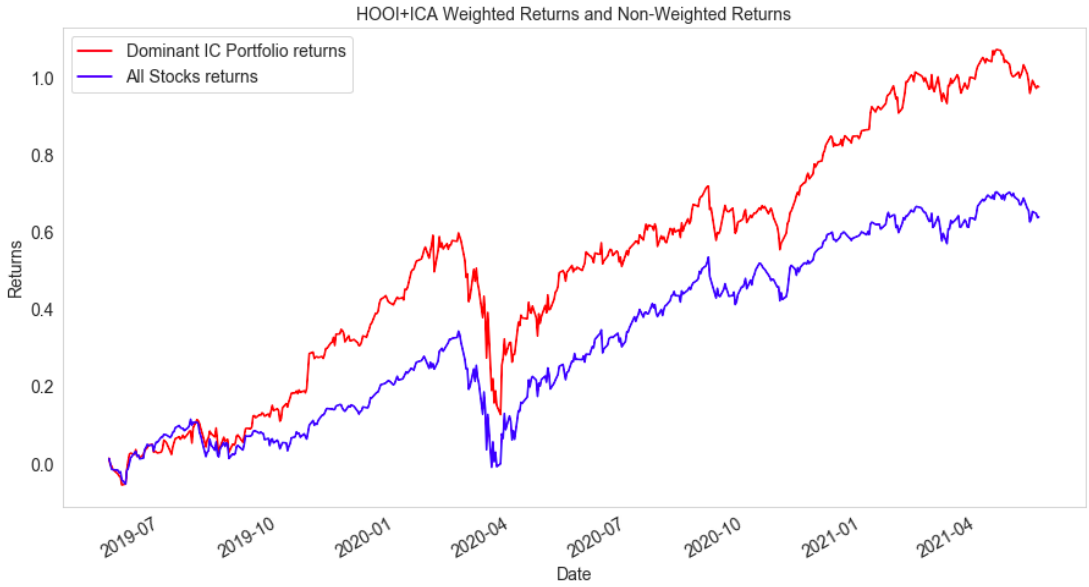

Figure 5.4.8: HOOI+ICA-weighted portfolio returns and unweighted stocks returns

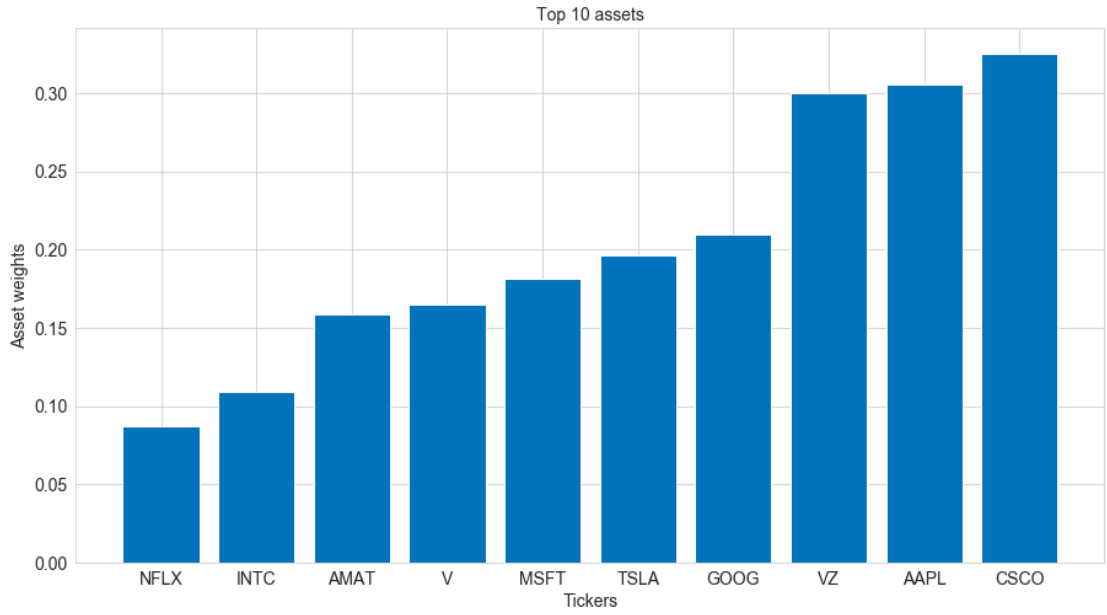

Figure 5.4.9: HOOI+ICA top 10 stocks

5.4.3 Performance Analysis

As observed from the tables 5.4.1-5.4.4, ICA-weighted portfolio gives the most favourable results in terms of annual return (51.331%), cumulative returns (129.012%), annual volatility (38.207%),sharpe ratio (1.28) and skew (-1.21). The PCA-weighted portfolio showed the least favourable values compared to other portfolios in terms of all performance metrics except maximum drawdown (-30.361%) and kurtosis (8.69), which were the best values among all portfolios. It can also be concluded that portfolios based on HOOI algorithm have consistently performed better than HOSVD-based and CPD-based portfolios.

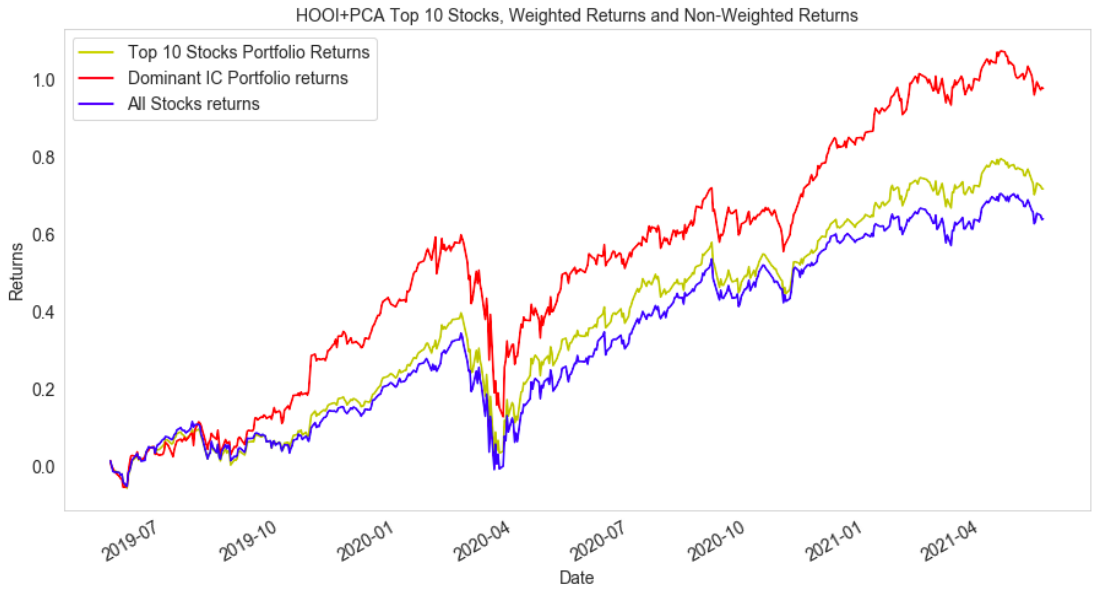

Figure 5.4.10: HOOI+ICA top 10 stocks portfolio returns, HOOI+ICA-weighted portfolio returns and unweighted stocks returns

Metric	Value
Annual return	36.541%
Cumulative returns	86.434%
Annual volatility	30.619%
Sharpe ratio	1.17
Maximum Drawdown	-33.056%
Skew	-1.13
Kurtosis	10.96

Table 5.4.4: HOOI+ICA top 10 stocks portfolio performance

CHAPTER 6

Results Evaluation

Metric	Non-weighted	PCA	ICA	CPD+PCA	CPD+ICA	HOSVD+PCA	HOSVD+ICA	HOOI+PCA	HOOI+ICA
Annual return	31.35%	39.698%	102.408%	55.688%	6.371%	22.314%	24.222%	27.313%	51.331%
Cumulative returns	72.527%	95.156%	309.691%	143.453%	13.207%	49.607%	54.312%	62.085%	129.012%
Annual volatility	30.513%	33.344%	50.589%	37.148%	4.647%	31.468%	25.086%	29.769%	38.207%
Sharpe ratio	1.05	1.17	1.65	1.37	1.30	0.80	0.99	0.96	1.28
Maximum Drawdown	-32.197%	-34.044%	-43.127%	-37.795%	-5.037%	-31.359%	-25.79%	-30.361%	-41.088%
Skew	-0.96	-0.97	-0.08	-0.92	-0.69	-0.75	-0.33	-0.88	-1.21
Kurtosis	9.35	7.87	3.96	6.08	5.136	8.10	1.20	8.69	13.81

Table 6.0.1: Weighted portfolio performance

Comparing the performance metrics for various methods from Table 6.0.1, Figure 6.0.1 and Figure 6.0.2, the best weighted portfolio results were obtained by applying Independent Component Analysis on the returns data. It had the highest annual and cumulative returns, annual volatility, sharpe ratio, and low kurtosis, which are preferred to construct a portfolio. However, its maximum drawdown has been the highest and skew has been the lowest among all the other methods. Although high volatility can lead to significant investment risks, when correctly harnessed, it can also generate high returns. Among the tensor methods, CPD+PCA method demonstrated the highest returns, volatility and sharpe ratio, and favourable skew and kurtosis values. However, its value for maximum drawdown was not preferred. Moreover, as discussed in Chapter 5, the performance of the CPD method is inconsistent which makes it unreliable.

Metric	Non-weighted	PCA	ICA	CPD+PCA	CPD+ICA	HOSVD+PCA	HOSVD+ICA	HOOI+PCA	HOOI+ICA
Annual return	31.35%	46.763%	40.18%	32.815%	38.875%	22.491%	40.845%	37.719%	36.541%
Cumulative returns	72.527%	115.393%	96.505%	77.822%	93.269%	50.039%	98.373%	89.664%	86.434%
Annual volatility	30.513%	32.043%	31.891%	31.635%	33.403%	32.462%	30.359%	33.879%	30.619%
Sharpe ratio	1.05	1.25	1.22	1.03	1.15	0.79	1.28	1.12	1.17
Maximum Drawdown	-32.197%	-35.424%	-32.546%	-33.128%	-33.858%	-32.383%	-33.726%	-33.678%	-33.056%
Skew	-0.96	-0.96	-1.02	-0.81	-0.92	-0.78	-1.28	-0.90	-1.13
Kurtosis	9.35	6.00	7.12	8.32	7.79	8.89	10.29	8.76	10.96

Table 6.0.2: Top 10 stocks portfolio performance

Comparing the performance metrics for various methods from Table 6.0.2, Figure 6.0.3 and Figure 6.0.4, the best top 10 stocks portfolio results were obtained by applying Principal Component Analysis on the returns data. It had the highest annual and cumulative returns, high sharpe ratio and skewness, and the lowest kurtosis. However, the annual volatility was low and the maximum drawdown was least favourable. Contrary to the previous case where more risk led to higher returns, low volatility stocks had outperformed high volatility stocks to give higher returns. This is called low volatility anomaly, which has been shown in recent research [76][77][78]. Among tensor methods, HOSVD+ICA gave the highest returns, sharpe ratio and skew, but the lowest annual volatility value. However, the values for maximum drawdown and kurtosis were among the least favourable.

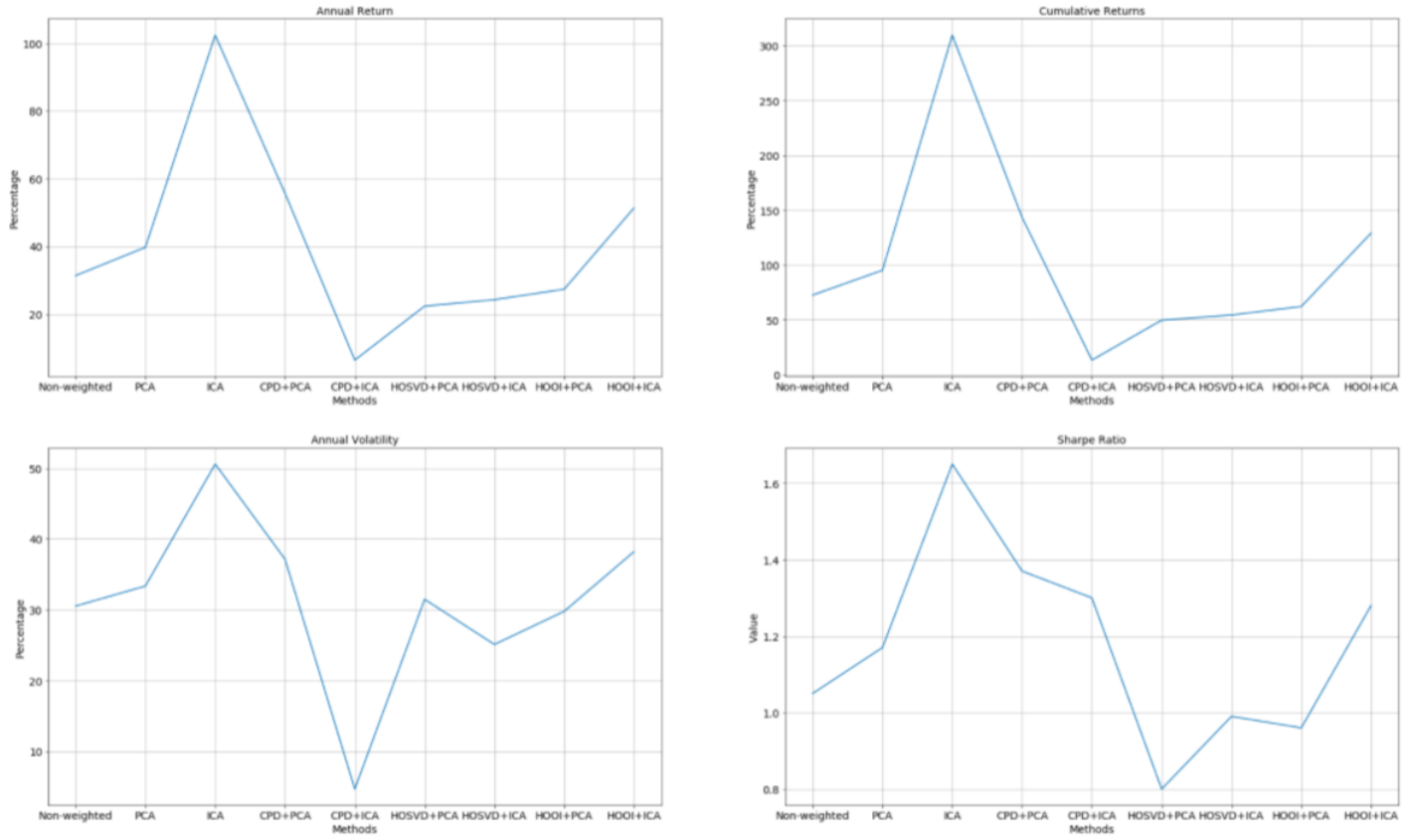

Figure 6.0.1: Performance evaluation for weighted portfolio: Annual returns, cumulative returns, annual volatility and sharpe ratio

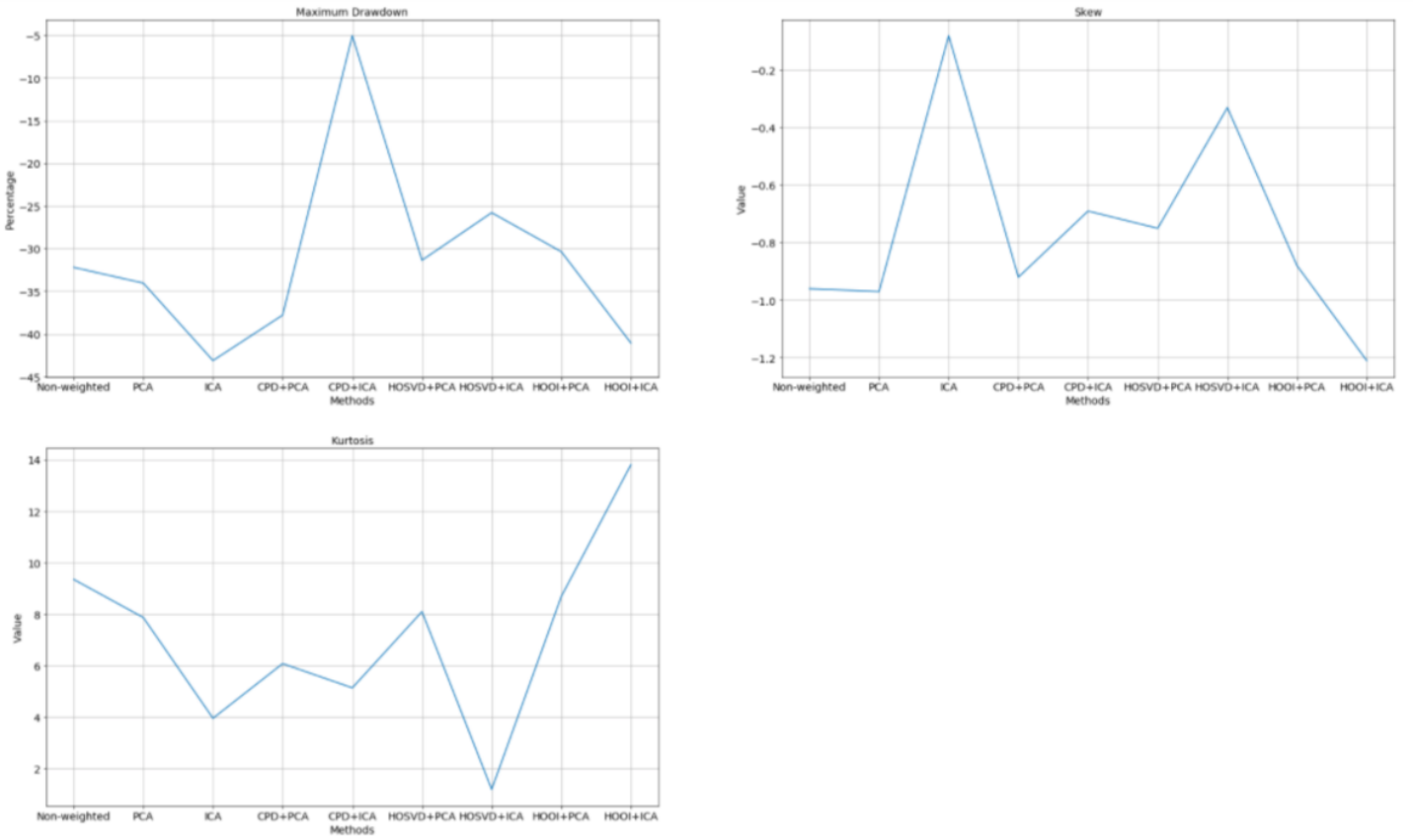

Figure 6.0.2: Performance evaluation for weighted portfolio: Maximum drawdown, skew and kurtosis

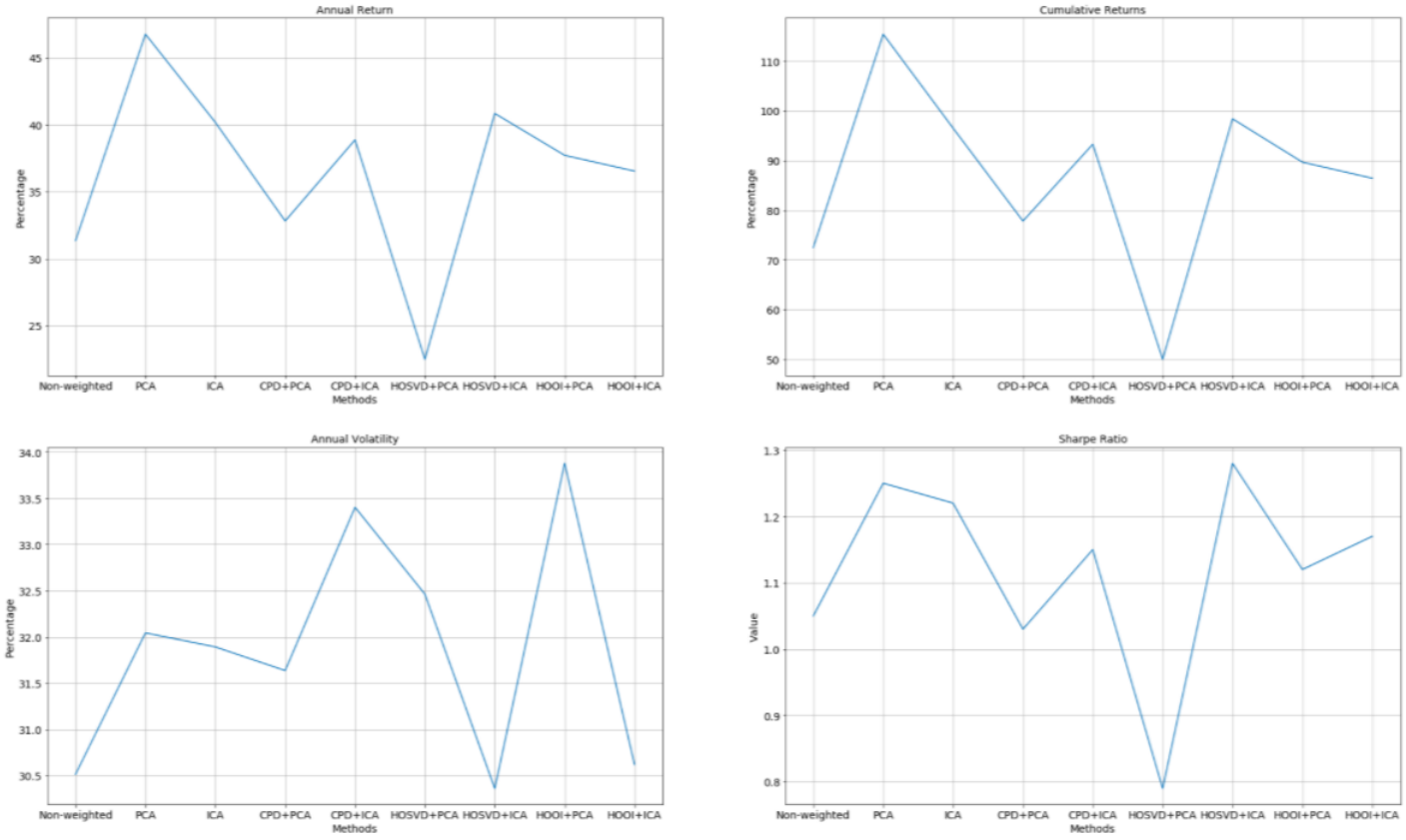

Figure 6.0.3: Performance evaluation for top 10 stocks portfolio: Annual returns, cumulative returns, annual volatility and sharpe ratio

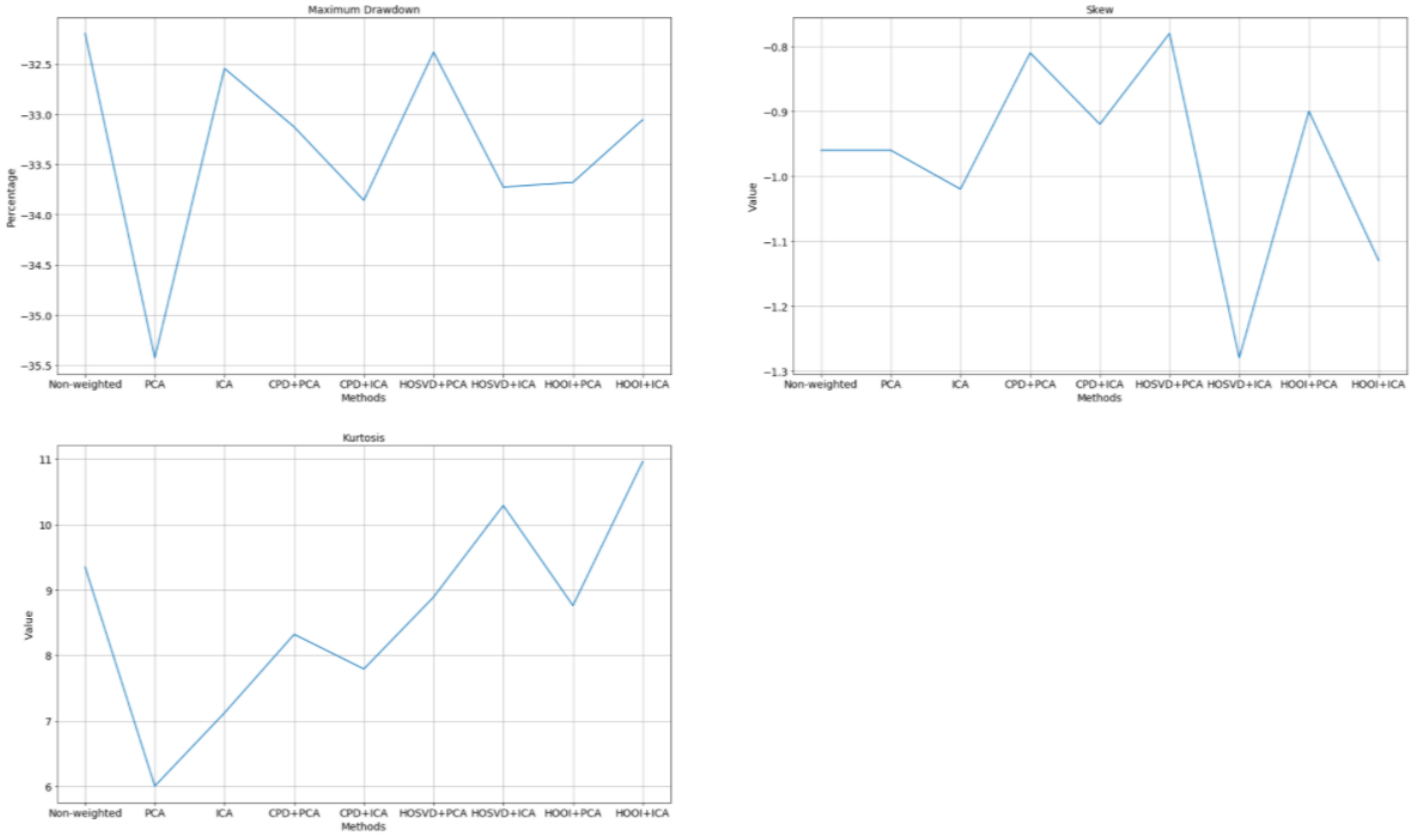

Figure 6.0.4: Performance evaluation for top 10 stocks portfolio: Maximum drawdown, skew and kurtosis

CHAPTER 7

Conclusion and Future Work

Tensor decomposition methods demonstrate the ability to combine stocks market data and fundamentals data and use statistical analysis on the assets feature matrix to find the dominant or weighted assets. Tensorization of multi-dimensional financial data has shown decent success in analysing financial data and determining optimal investment portfolio.

However, the conventional ICA and PCA on the returns data showed the best results for the weighted portfolio and the portfolio containing the top 10 highest weighted stocks respectively. The performance results obtained from certain tensor-based methods for constructing the top 10 stocks portfolios were close to the results obtained by simply applying statistical analysis on the returns data.

In the case of weighted portfolio, ICA had outperformed tensor methods by a wide margin. Combined tensor and statistical methods, such as CPD+PCA and HOOI+ICA methods achieved an annual return of 55.688% and 51.311% respectively, which were nearly half the annual return obtained from ICA (102.408%). Moreover, both tensor methods demonstrated high sharpe ratio, but the value was much higher for ICA. (1.37 for CPD+PCA, 1.28 for HOOI+ICA and 1.65 for ICA).

In the case of top 10 selected stocks portfolio, although PCA achieved the highest annual return (46.763%), combined tensor and statistical methods, such as HOSVD+ICA (40.845%), CPD+ICA (38.875%), HOOI+PCA (37.719%) and HOOI+ICA (36.541%) demonstrated high values for annual returns as well. Moreover, HOSVD+PCA method showed a slightly higher sharpe ratio compared to PCA. (1.28 for HOSVD+ICA against 1.25 for PCA).

The most challenging parts of the project were to find a method for extracting the weighted or dominant assets from the decomposed matrices and accommodating for inconsistent results from Canonical Polyadic Decomposition. Since it is not possible to qualitatively observe the performance of the assets from the decomposed feature matrices, tensor methods had to be combined with statistical analysis methods such as ICA and PCA to obtain the best assets for the portfolio. Moreover, it was also observed that CPD showed less consistent results (Table 5.2.1-5.2.4) compared to Tucker Decomposition algorithms (HOSVD and HOOI). As a solution to this inconsistent nature of CPD, the mean value for 5 trials of the performance metrics was considered. However, this solution is not completely reliable due to high deviation in the values.

7.0.1 Future Work

Overall, the project has met its aim to use tensor methods to combine stocks market data and financial fundamentals data and optimise portfolio construction. This is definitely a breakthrough in financial signal processing for portfolio optimisation. A new method has been developed to combine different stocks data together for analysis rather than simply applying conventional statistical analysis on returns data. Several tensor methods when

combined with statistical analysis methods demonstrated intuitive results, which have been discussed in earlier chapters. However, testing the financial implementation on large and diversified set of stocks data and improving the implementation through research can enhance the performance of tensor methods over conventional statistical analysis methods.

Financial implementation in this project has been restricted to just 20 technology stocks listed on NYSE. Applying the implementation on more number of stocks belonging to a range of sectors and including more stocks data can help gain more intuition on the performance of the tensor-based approach.

CPD and Tucker-based tensor decomposition algorithms have been explored in this project. Although the performance of CPD has been promising for some portfolios, the inconsistency in results can encourage us to explore other well-established tensor decomposition methods such as Tensor Train Decomposition and observe whether the results improve and remain consistent.

In this project, data is tensorized into a 3D tensor such that each mode represents time steps (dates), assets and stocks data. However, other methods for tensorizing stocks data can be explored. A correlation-based approach for tensorizing stocks data has been explained in [74] and a three-way tensor representation has been explained in [75].

The trading performance for tensor methods varies greatly with rank. The kruksal rank for the CPD and the multi-linear rank for Tucker Decomposition were fixed to 100 and (2010,20,7) respectively and the project does not explore the performance of the tensor methods for different rank values. Model selection can be proposed to run the algorithms for different ranks and choose a value which gives the best performance metrics for the particular tensor decomposition method. This can help achieve better performance than the conventional statistical analysis techniques such as PCA and ICA.

On the basis of the results of this study, it is clear that tensors provide a formal framework for financial analysis which can lead to impressive breakthroughs in various financial applications including portfolio construction. In contrast to the two-dimensional matrix analysis, tensor methods can process high volume, variety and complexity of stocks data together which can help identify key performance trends and factors for constructing optimal portfolios. With further research in this domain, it is possible to achieve results that can outperform the conventional methods.

Bibliography

[1] Y. Ji, Q. Wang, X. Li, and J. Liu. A survey on tensor techniques and applications in machine learning. *IEEE Access*, 7:162950–162990, 2019.

[2] Andrzej Cichocki. Tensor networks for big data analytics and large-scale optimization problems. *CoRR*, abs/1407.3124, 2014.

[3] C. Caiafa A.-H. Phan G. Zhou Q. Zhao A. Cichocki, D. Mandic and L. D. Lathauwer. Multiway component analysis: Tensor decompositions for signal processing applications. *IEEE Signal Processing Magazine*, 2014.

[4] A. Cichocki. Era of big data processing: A new approach via tensor networks and tensor decompositions. 2014.

[5] A.-H. Phan A. Cichocki, R. Zdunek and S. Amari. Nonnegative matrix and tensor factorizations: Applications to exploratory multi-way data analysis and blind source separation. *Chichester: Wiley*, 2009.

[6] T. Kolda and B. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

[7] A. Cichocki. Tensors decompositions: New concepts for brain data analysis? *Journal of Control, Measurement, and System Integration (SICE)*, 47(7):507–517, 2011.

[8] W. Hackbusch. Tensor spaces and numerical tensor calculus. *ser. Springer series in computational mathematics. Heidelberg: Springer*, 42, 2012.

[9] R. Bro A. Smilde and P. Geladi. Multi-way analysis: Applications in the chemical sciences. *New York: John Wiley Sons Ltd*, 2014.

[10] P. Kroonenberg. Applied multiway data analysis. *New York: John Wiley Sons Ltd*, 2008.

[11] M. Agarwal V. Bhasker S. Sehgal, H. Singh and Shantanu. Data analysis using principal component analysis. *2014 International Conference on Medical Imaging, m-Health and Emerging Communication Systems (MedCom), Greater Noida*, pages 45–48, 2014.

[12] IPING TU HUNG HUNG, PEISHIEN WU and SUYUN HUANG. On multilinear principal component analysis of order-two tensors. *Biometrika*, 99(3):569–583, 2012.

[13] G. Pasini. Principal component analysis for stock portfolio management. *International Journal of Pure and Applied Mathematics*, 115(1):153–167, 2017.

[14] Libin Yang. An application of principal component analysis to stock portfolio management. *Department of Economics and Finance, University of Cantebury*, 2015.

[15] Andrew D. Back and Andreas S. Weigend. A first application of independent component analysis to extracting structure from stock returns, 1997.

[16] Siu-Ming CHA and Lai-Wan CHANn. Applying independent component analysis to factor model in finance, 2000.

[17] Dinesh Kant Kumar Djuwari Djuwari and Marimuthu Palaniswami. Limitations of ica for artefact removal, 2005.

[18] Shawhin Talebi. Independent component analysis (ica).

[19] Xiyuan Zhang Kaiqi Zhang and Zheng Zhang. Tucker tensor decomposition on fpga. 2017.

[20] T. Kolda and B. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

[21] C. Caiafa A.-H. Phan G. Zhou Q. Zhao A. Cichocki, D. Mandic and L. D. Lathauwer. Multiway component analysis: Tensor decompositions for signal processing applications. *IEEE Signal Processing Magazine*, 2014.

[22] Debals and L. De Lathauwer. Stochastic and deterministic tensorization for blind signal separation. *Proceedings of the 12th International Conference Latent Variable Analysis and Signal Separation, Springer International Publishing*, 322(10):3–13, 2015.

[23] Q. Zhao-N. Lee I. V. Oseledets M. Sugiyama A. Cichocki, A-H. Phan and D. Mandic. Tensor networks for dimensionality reduction and large-scale optimizations. part 2 applications and future percepectives. *arXiv e-prints*, art. arXiv:1708.09165, 2019.

[24] Boris N. Khoromskij. Fast and accurate tensor approximation of a multivariate convolution with linear scaling in dimension. *Journal of Computational and Applied Mathematics*, 234(11):3122 – 3139, 2010.

[25] J. Hastad. Tensor rank is np-complete. *Journal of Algorithms*, 11(4):644–654, 1990.

[26] R. Bro A. Smilde and P. Geladi. Multi-way analysis: Applications in the chemical sciences. *New York: John Wiley Sons Ltd*, 2004.

[27] Mark A. Iwen Ali Zare, Alp Ozdemir and Selin Aviyente. extension of pca to higher order data structures: an introduction to tensors, tensor decompositions, and tensor pca. *arXiv e-prints*, art. arXiv:1803.00704v2, 2018.

[28] R. A. Harshman. Foundations of the parafac procedure: Models and conditions for an explanatory multimodal factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84, 1970.

[29] Eric Moreau R. Andre, Xavier Luciani. A fast algorithm for the cp decomposition of large tensors. *SIS2017 STATISTICS AND DATA SCIENCE: NEW CHALLENGES, NEW GENERATIONS*, 322(10):891–921, 2017.

[30] Luciani X. de Almeida A.L.F Comon, P. Tensor decompositions, alternating least squares and other thales. *Journal of Chemometrics*, 23, 2009.

[31] Dunlavy D.M. Kolda T.G Acar, E. A scalable optimization approach for fitting canonical tensor decompositions. *Journal of Chemometrics*, 25(2):67–86, 2011.

[32] Jianguang Zhang, Yahong Han, and Jianmin Jiang. Tucker decomposition-based tensor learning for human action recognition. *Multimedia Systems*, 22, 2015.

[33] L. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[34] B. De Moor L. De Lathauwer and J. Vandewalle. On the best rank-1 and rank-(r1, r2,..., rn) approximation of higher-order tensors. *SIAM J. Matrix Analysis and Applications*, 21(4):1324–1342, 2000.

[35] P. M. Kroonenberg and J. De Leeuw. Principal component analysis of three-mode data by means of alternating least squares algorithms. *Psychometrika*, 45(1):69–97, 1980.

[36] H. Neudecker A. Kapteyn and T. Wansbeek. An approach ton-mode components analysis. *Psychometrika*, 51(2):269–275, 1986.

[37] Lieven De Lathauwer and Joos Vandewalle. Dimensionality reduction in higher-order signal processing and rank-(r1,r2,,rn) reduction in multilinear algebra. *Linear Algebra and its Applications*, 391:31 – 55, 2004.

[38] J. G. Nagy and M. E. Kilmer. Kronecker product approximation for pre- conditioning in three-dimensional imaging applications. *IEEE Transactions on Image Processing*, 15(3):604–613, 2006.

[39] Morten Mrup. Applications of tensor (multiway array) factorizations and de- compositions in data mining. *Wiley Interdisc. Rew.: Data Mining and Knowl- edge Discovery*, 1(1):24–40, 2011.

[40] Jun Yan Zheng Chen Wenyin Liu Fengshan Bai Ning Liu, Benyu Zhang and Leefeng Chien. Text representation: from vector to tensor. *In Fifth IEEE International Conference on Data Mining (ICDM'05)*, 2005.

[41] Berkant Savas and Lars Eldn. Handwritten digit classification using higher order singular value decomposition. *Pattern Recognition*, 40(3):993 – 1003, 2007.

[42] M. A. O. Vasilescu and D. Terzopoulos. Multilinear analysis of image ensembles: Tensorfaces. *Proceedings of the 7th European Conference on Computer Vision, Lecture Notes in Computer Science, Springer, New York*, 2350:447–460, 2002.

[43] L. Shi Y. Yu H. Wang, Q. Wu and N. Ahuja. Out-of-core tensor approximation of multidimensional matrices of visual data. *ACM Transactions on Graphics (Special Issue for SIGGRAPH 2005)*, 24:527–535, 2005.

[44] R. Henrion. Simultaneous simplification of loading and core matrices in n-way pca: Application to chemometric data arrays. *Fresenius' Journal of Analytical Chemistry*, 361:15–22, 1998.

[45] R. Henrion. On global, local and stationary solutions in three-way data analysis. *J. Chemometrics*, 14:261–274, 2000.

[46] I. Oseledets A. H. Phan Q. Zhao D. Mandic et al. A. Cichocki, N. Lee. Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions. *Foundations and Trends in Machine Learning*, 9(4-5):249–429, 2016.

[47] R. Or´us. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of Physics*, 349:117–158, 2014.

[48] L. Grasedyck. Hierarchical singular value decomposition of tensors. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2029–2054, 2010.

[49] D. Kressner L. Grasedyck and C. Tobler. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitteilungen*, 36(1):53–78, 2013.

[50] W. Hackbusch and S. K¨uhn. A new scheme for the tensor representation. *Journal of Fourier analysis and applications*, 15(5):706–722, 2009.

[51] I. V. Oseledets. Tensor-train decomposition. *SIAM J. Sci. Comput.*, 33(5):2295–2317, 2011.

[52] Ivan V Oseledets and SV Dolgov. Solution of linear systems and matrix inversion in the tt-format. *SIAM Journal on Scientific Computing*, 34(5):A2718–A2739, 2012.

[53] Ivan V Oseledets George E Karniadakis Zheng Zhang, Xiu Yang and Luca Daniel. Enabling high-dimensional hierarchical uncertainty quantification by anova and tensor-train decomposition. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(1):63–76, 2014.

[54] Luca Daniel Haotian Liu and Ngai Wong. Model reduction and simulation of nonlinear circuits via tensor decomposition. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(7):1059–1069, 2015.

[55] Haotian Liu Luca Daniel Zheng Zhang, Kim Batselier and Ngai Wong. Tensor computation: A new framework for high-dimensional problems in eda. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 36(4):521–536, 2017.

[56] Zhongming Chen Kim Batselier and Ngai Wong. Tensor network alternating linear scheme for mimo volterra system identification]. *Automatica*, 84:26–35, 2017.

[57] Ivan V Oseledets. Approximation of 2d × 2 d matrices using tensor decomposition. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2130– 2145, 2010.

[58] Daniel Kressner and Andr´e Uschmajew. On low-rank approximability of solutions to high-dimensional operator equations and eigenvalue problems. *Linear Algebra and its Applications*, 493:556–572, 2016.

[59] Luca Daniel Kim Batselier, Wenjian Yu and Ngai Wong. Computing low-rank approximations of large-scale matrices with the tensor network randomized svd. *SIAM Journal on Matrix Analysis and Applications*, 39(3):1221–1244, 2018.

[60] P. A. Wilkinson. Application of the tensor train decomposition in machine learning - a study and tradeoffs. *Dissertation, University at Buffalo, State University of New York*, 2019.

[61] P. Gelß. The tensor-train format and its applications: Modeling and analysis of chemical reaction networks, catalytic processes, fluid flows, and brownian dynamics. *Dissertation, Freie Universit€at Berlin, Berlin*, 2017.

[62] S. Xie L. Zhang Q. Zhao, G. Zhou and A. Cichocki. Tensor ring decomposition. *arXiv preprint*, arXiv:1606.05535, 2016.

[63] V. Aggarwal W. Wang and S. Aeron. Efficient low rank tensor ring completion. *Rn*, 1(r1):1, 2017.

[64] Corporate Finance Institute. Asset classes.

[65] Akhilesh Ganti. Asset class.

[66] Angelos Filos. Reinforcement learning for portfolio management. *arXiv e-prints*, art. arXiv:1909.09571, 2018.

[67] Charles River. Enhancing portfolio construction with factor models.

[68] Ilya Kisil. Hottbox: Higher order tensors toolbox.

[69] Investopedia. Momentum indicates stock price strength.

[70] investyadnya. Important factors in fundamental analysis.

[71] LabsterX. How to do fundamental analysis on stocks using yahoo! finance.

[72] Yao Lei Xu. Tensor-train recurrent neural networks in financial forecasting, 2019.

[73] Cathy O'Neil. Why log returns.

[74] A. Spelta. Financial market predictability with tensor decomposition and links forecast, 2017.

[75] Li Ling Jiang Ping Li Qing Li, Yuanzhu Chen and Hsinchun Chen. A tensor-based information framework for predicting the stock market, 2016.

[76] Rashid Rasul. Why does it benefit to have low volatility stocks in your portfolio?

[77] Pim Van Vliet David Blitz and Guido Baltussen. The volatility effect revisited, 2019.

[78] Nardin L. Baker and Robert A. Haugen. Low risk stocks outperform within all observable markets of the world, 2012.

APPENDIX A

Ethical, Legal and Safety Plan

A.0.1 Ethical Issues

There are no major ethical or moral issues related to the project.

A.0.2 Legal Issues

The only legal issue that can potentially arise from this project is related to the use of data. The use of stock and equity fundamental data provided by Yahoo Finance and WDMS will be subject to its term and services. Therefore, the use of data will be restricted to research purposes only and not for any commercial activity and will not be distributed to other parties.

A.0.3 Safety Issues

Multiple safety issues are considered for this project which are discussed below:

- **Data infrastructure safety**: The data used for the project will be provided by reliable sources (e.g. Yahoo Finance and WDMS for financial data) and its use would not compromise any IT systems via the spreading of viruses or improper use of networks.
- **Electrical safety**: There is no electrical/electronic equipment involved in the project that constitutes an electrical hazard.
- **Physical safety**: There is no large or fast-moving objects involved in the project that can cause harm to myself or others.
- **Chemical safety**: There is no poisonous, irritant or allergenic material involved in the project.
- **Fire safety**: There is no material or equipment involved in the project that constitutes a fire hazard.
- **Biological safety**: There is no material involved in the project that can cause possible biological hazards.
- **Animal safety**: There are no animals involved in the project.
- **Study participant safety**: There are no study participants involved in the project.

APPENDIX B

Stocks Data used in Tensor-based Analysis

	Date	Asset	Returns	Volume	Predictive Factor	Non-Predictive Factor	Momentum	Price to book Value	Market Cap	P/E
0	2011-05-20 00:00:00	AAPL	-0.810748	4.16896	0.481235	-0.585197	-0.658513	-0.139992	0.70079	-0.0575831
1	2011-05-20 00:00:00	ADBE	-0.173145	-0.364517	0.26945	0.889325	0.762768	-0.33564	-0.730837	-0.0488149
2	2011-05-20 00:00:00	AMAT	-0.868855	-0.175676	1.2116	-1.11379	-1.31808	-0.623497	-0.820739	-0.13414
3	2011-05-20 00:00:00	AMD	-0.147891	-0.182383	0.935859	-0.865218	-0.668475	-0.093475	-0.789831	-0.0792015
4	2011-05-20 00:00:00	AMZN	-0.0705784	-0.362698	0.926078	-0.218245	-0.0547063	0.497923	-0.377633	0.187881
...	...	...	...	...	...	...	...	...	...	...
40235	2019-05-20 00:00:00	TSLA	-1.38069	0.981545	2.88553	-3.11508	-3.22319	-0.106688	-0.649447	-0.281058
40236	2019-05-20 00:00:00	TXN	-0.996915	-0.306641	1.32623	-1.6561	-1.65237	0.356096	-0.313561	-0.0580447
40237	2019-05-20 00:00:00	V	-0.220526	-0.344192	-0.781094	0.387177	0.00153138	0.50291	1.39006	0.0379388
40238	2019-05-20 00:00:00	VZ	0.744173	-0.125277	-0.862245	0.519055	0.443085	-0.244783	0.371986	-0.0698332
40239	2019-05-20 00:00:00	XLNX	-1.82688	-0.26683	2.60496	-2.05439	-2.45291	0.204628	-0.69502	-0.00492527

Table B.0.1: First and last 5 rows of stocks data

APPENDIX C

Raw Data

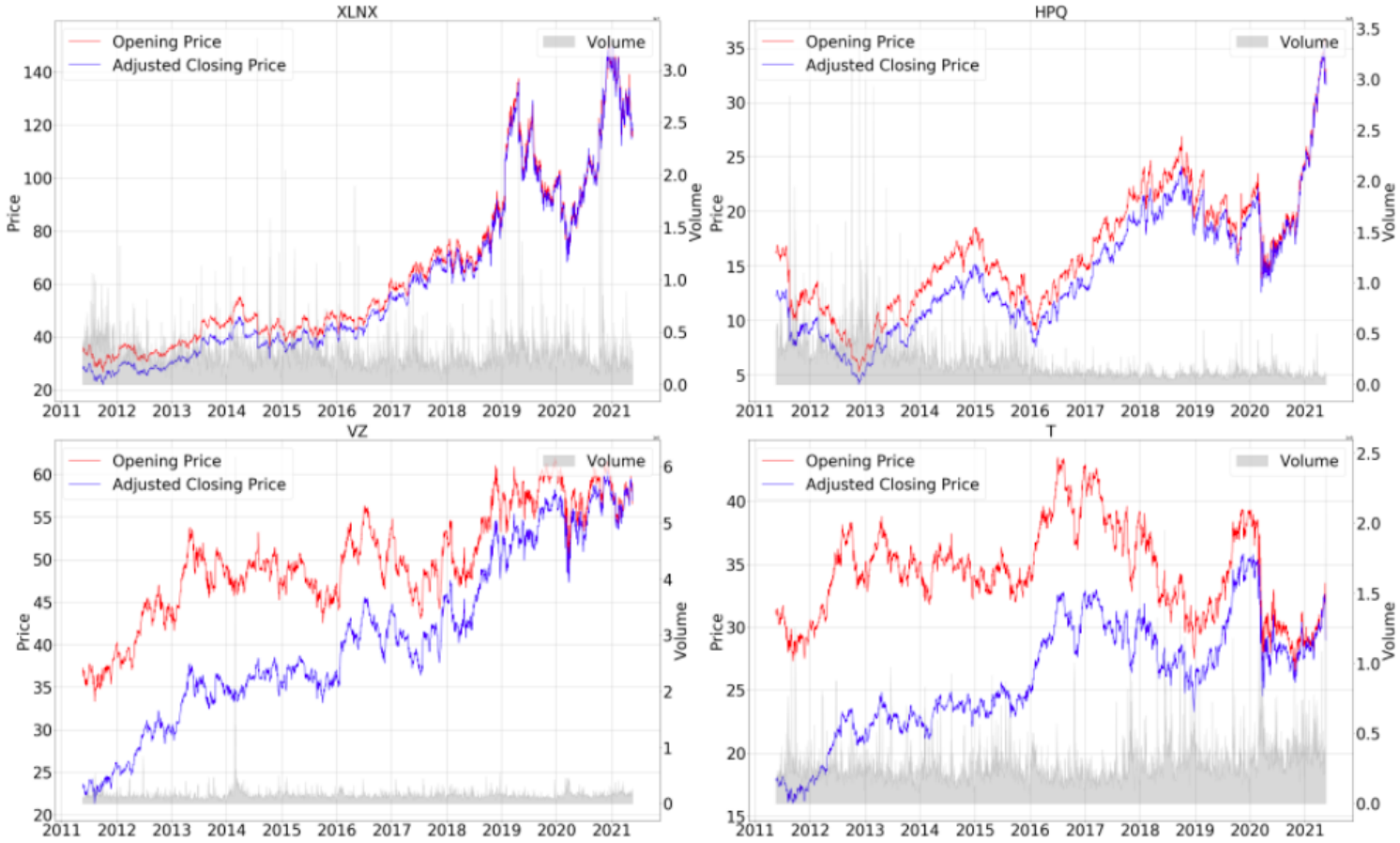

Figure C.0.1: Raw Price and Volume Data: XLNX, HPQ, VZ, T

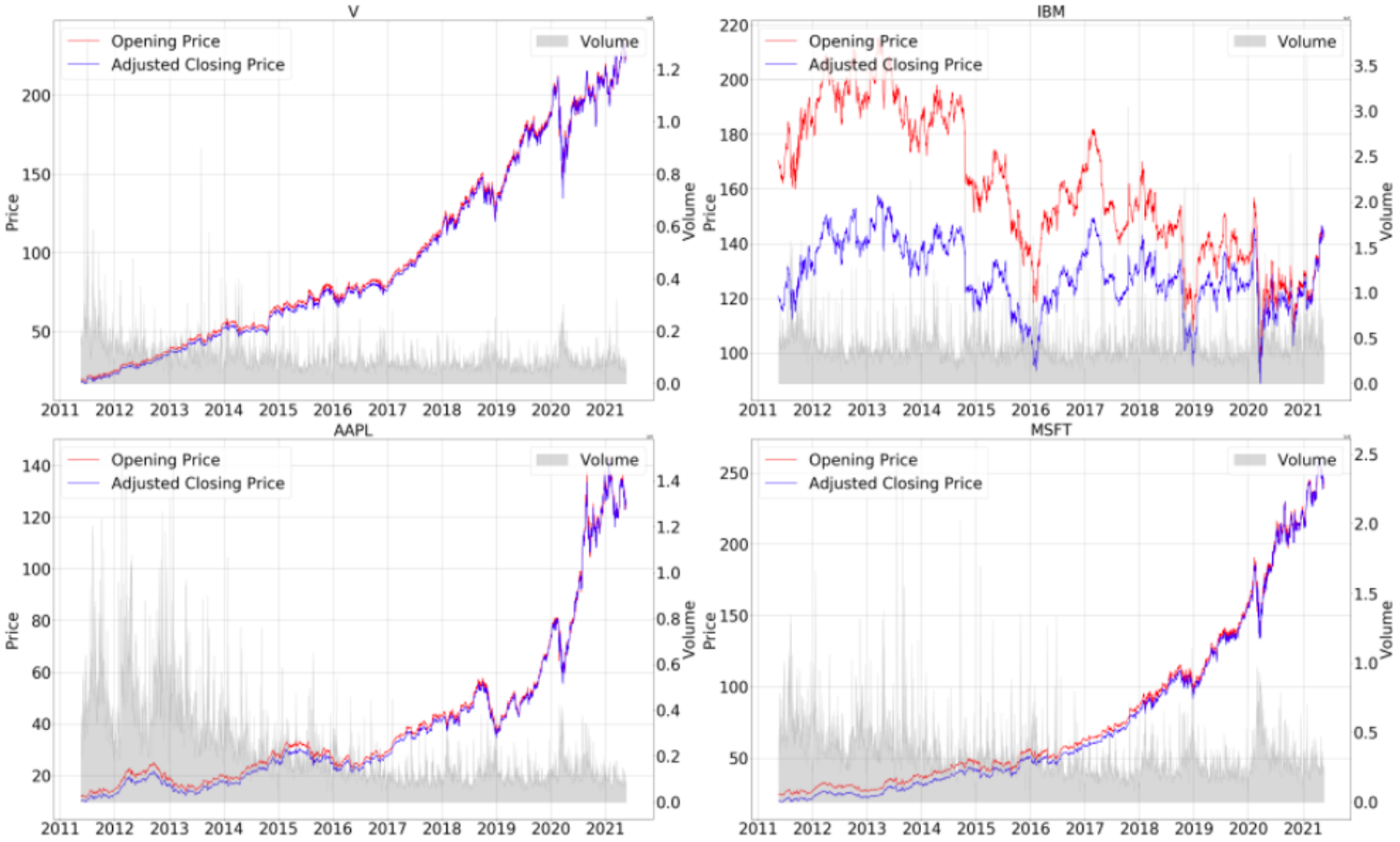

Figure C.0.2: Raw Price and Volume Data: V, IBM, AAPL, MSFT

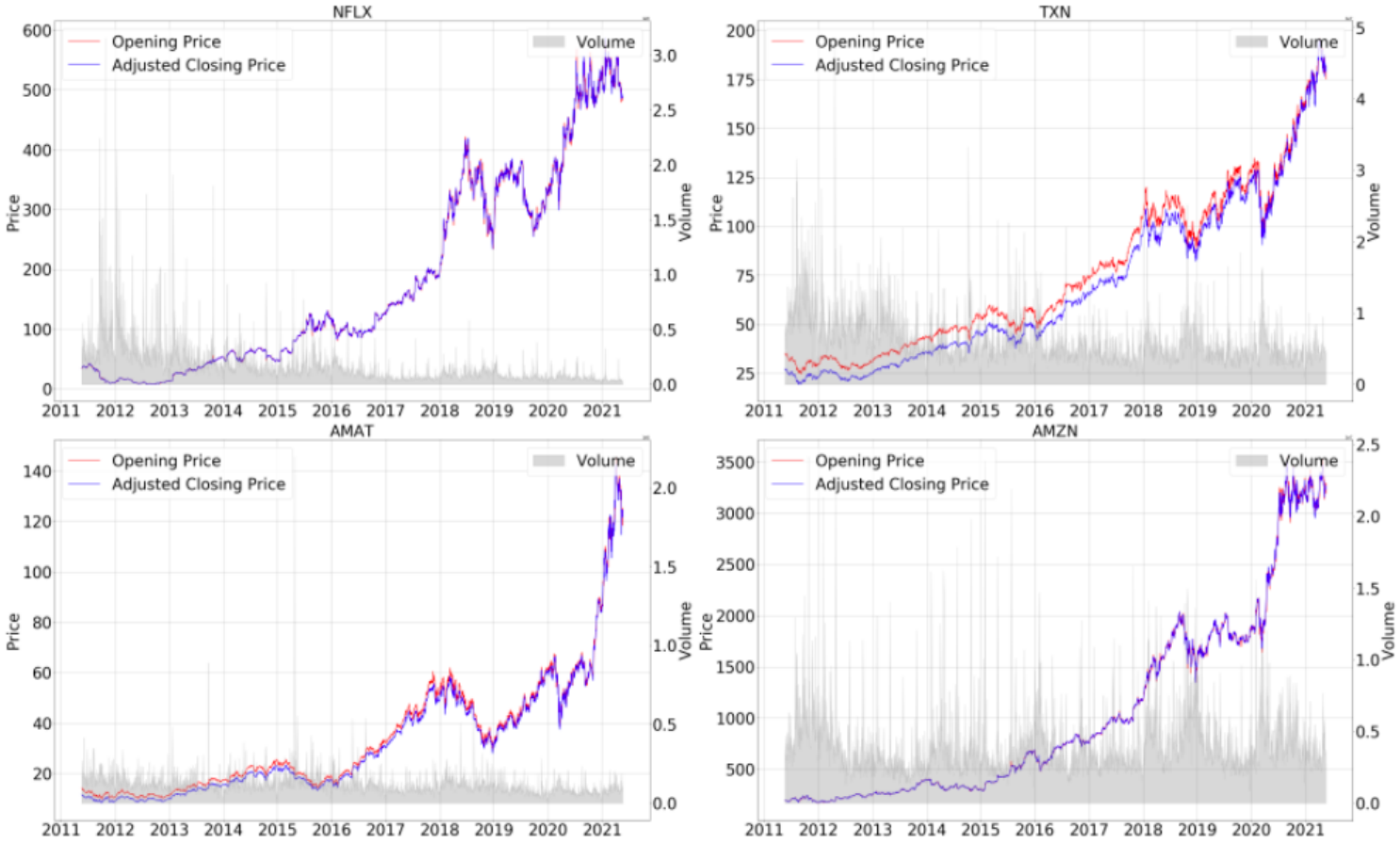

Figure C.0.3: Raw Price and Volume Data: NFLX, TXN, AMAT, AMZN

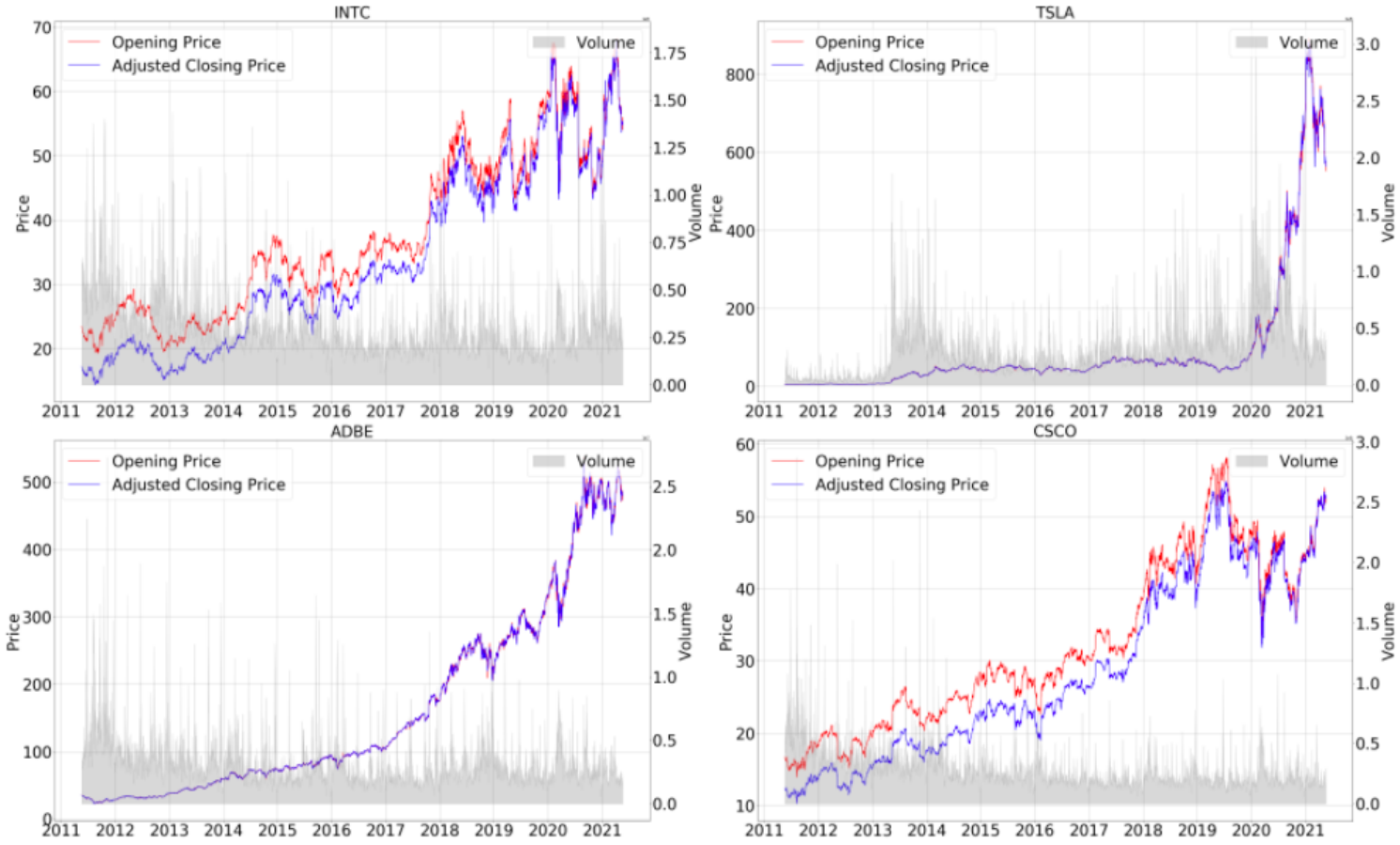

Figure C.0.4: Raw Price and Volume Data: INTC, TSLA, ADBE, CSCO

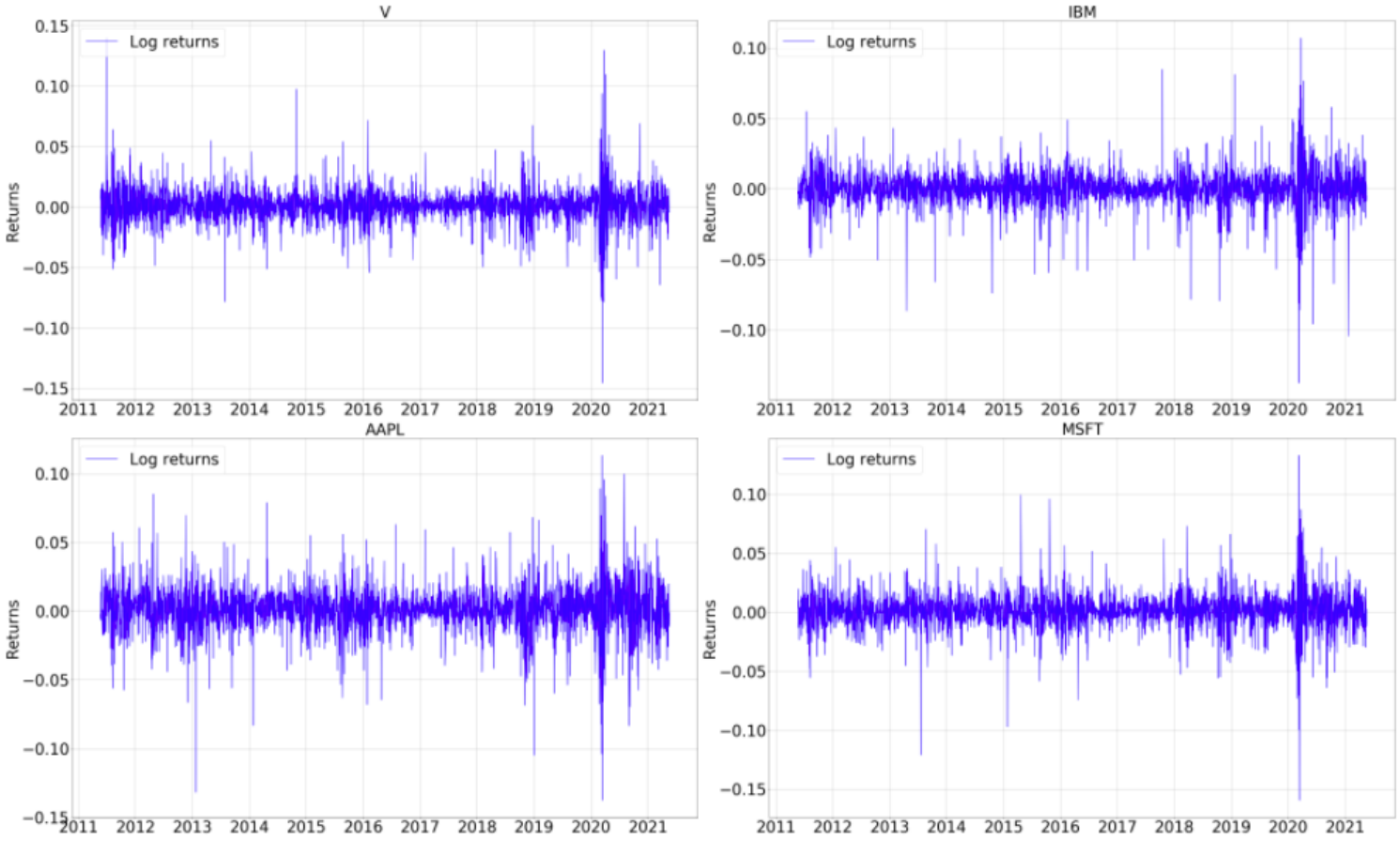

Figure C.0.5: Log Returns: V, IBM, AAPL, MSFT

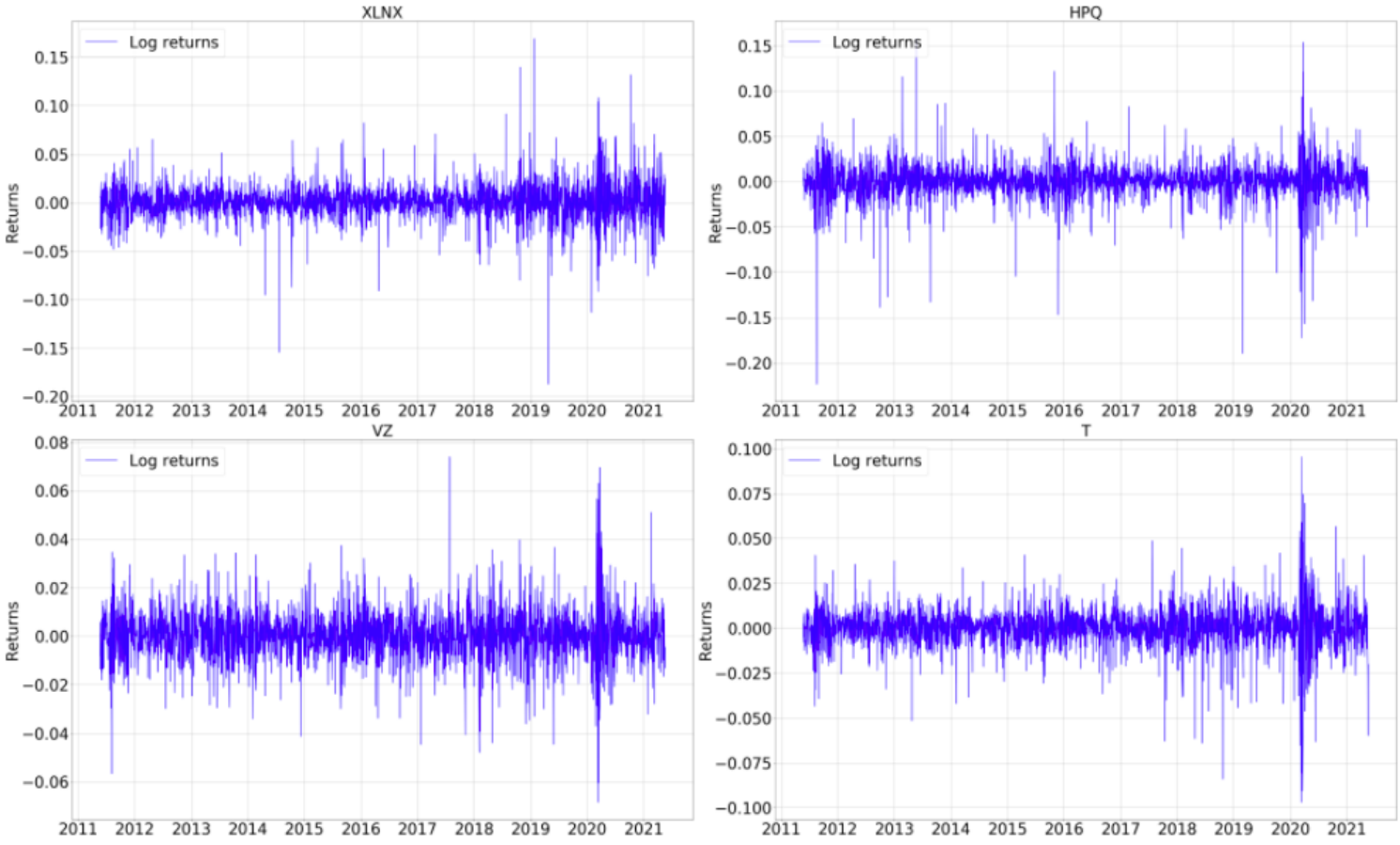

Figure C.0.6: Log Returns: XLNX, HPQ, VZ, T

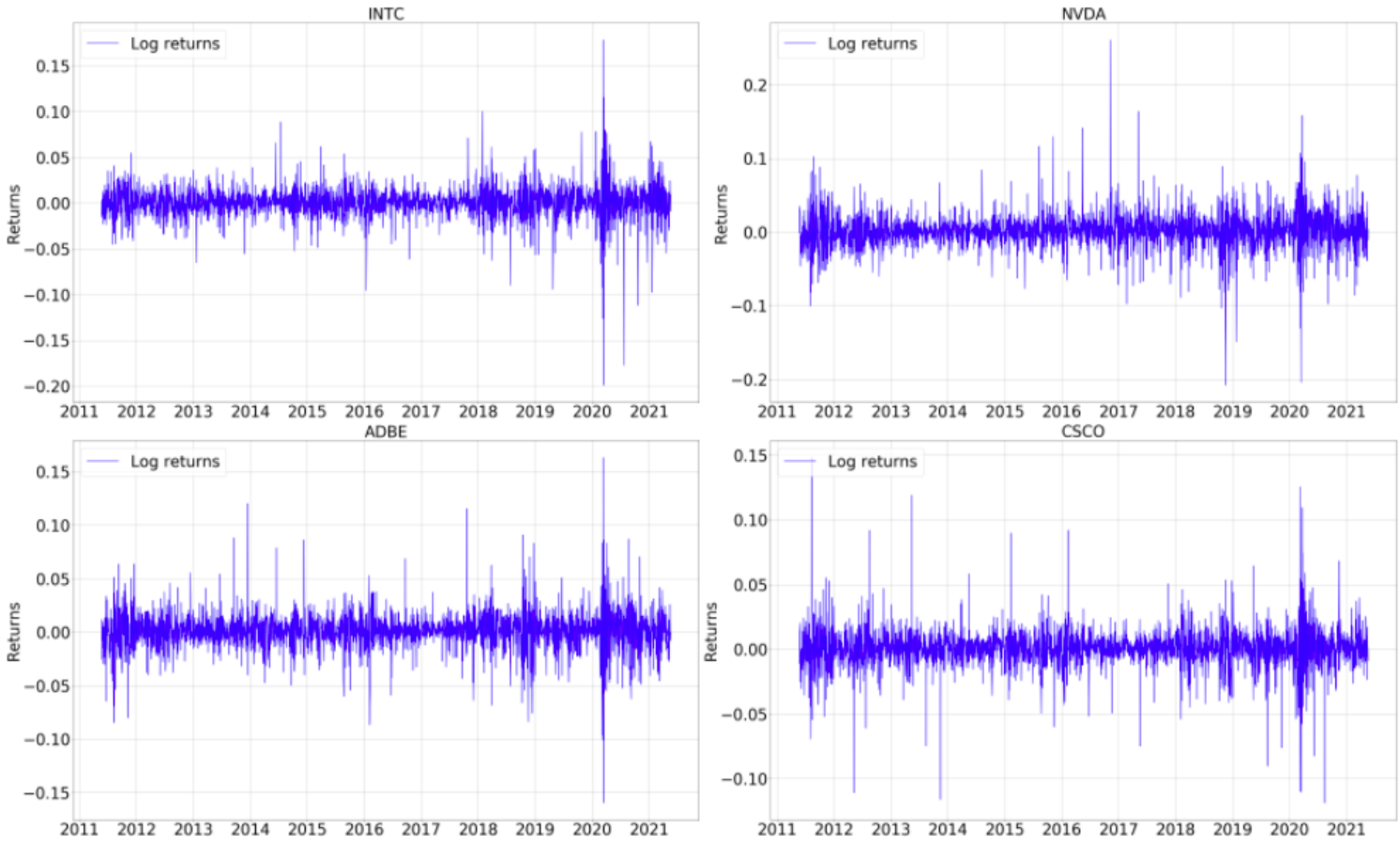

Figure C.0.7: Log Returns: INTC, NVDA, ADBE, CSCO

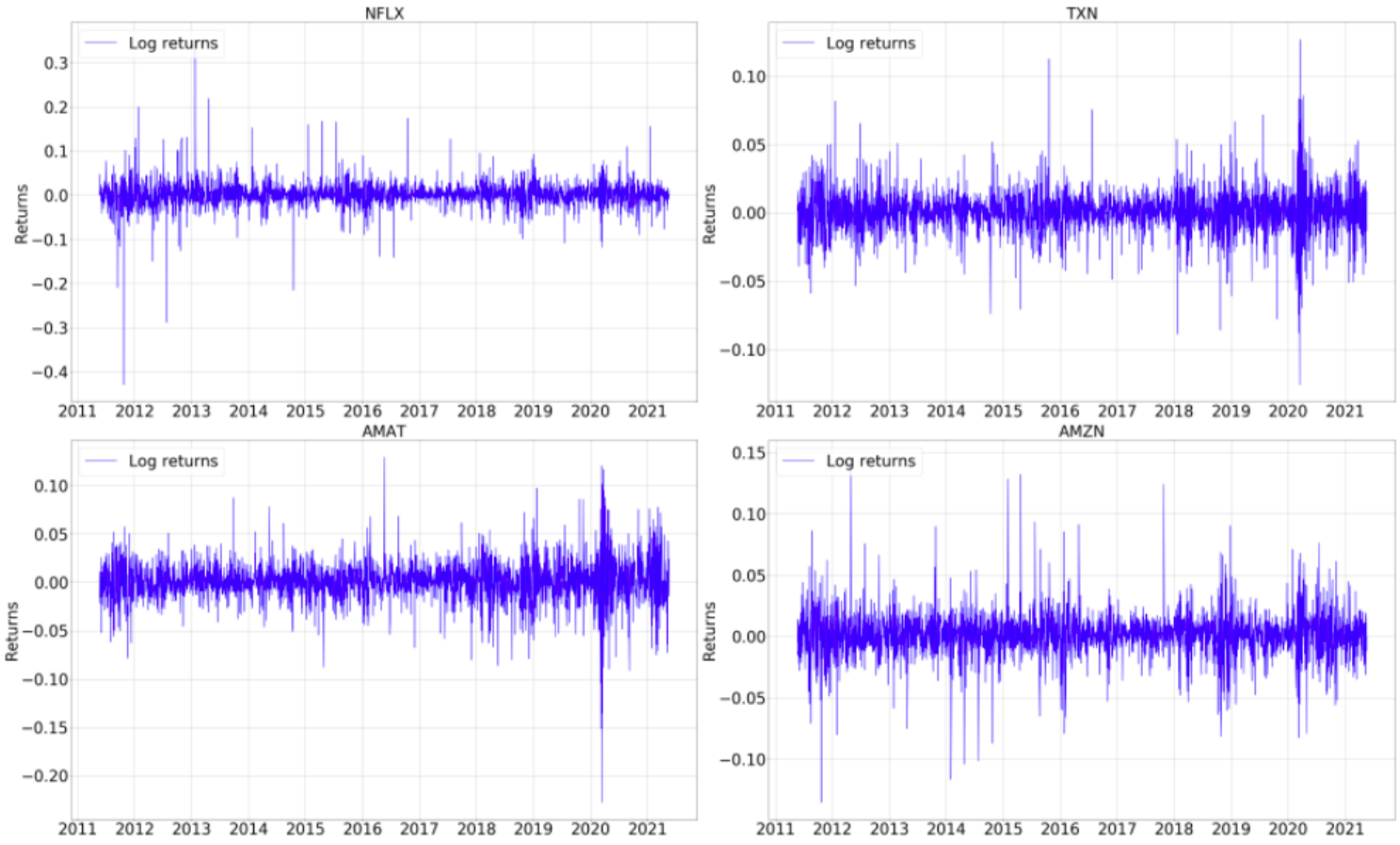

Figure C.0.8: Log Returns: NFLX, TXN, AMAT, AMZN

APPENDIX D

Code

The Jupyter notebooks for this project can be found on:
`https://github.com/snehil1998/Portfolio-Construction-using-Tensor-Methods`