# Deep neural network learning-based classifier design for big-data analytics

Krishnan Raghavan

# DEEP NEURAL NETWORK LEARNING-BASED CLASSIFIER DESIGN FOR BIG-DATA ANALYTICS

by

KRISHNAN RAGHAVAN

A DISSERTATION

Presented to the Graduate Faculty of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

COMPUTER ENGINEERING

2019

Approved by

## PUBLICATION DISSERTATION OPTION

This dissertation has been prepared using the publication option and is composed of five papers. Paper I, covers pages 15-51 and the title is *A Hierarchical Dimension Reduction Approach for Big Data with Applications to Fault Diagnostics*. This paper is under second revision with *Big Data Research*. Paper II covers pages 52-94 and the title is *A Multi-Step Nonlinear Dimension-Reduction Approach with Applications to Big Data*. This paper is published in *IEEE Transactions on Knowledge and Data Engineering*.

Paper III covers pages 95-122 and the title is *Direct Error Driven Learning for Deep Neural Networks in Applications Generating Big Data*. This paper is accepted with minor revision in *IEEE Transactions on Neural Networks and Learning Systems*. Paper IV covers pages 123-162 and the title is *A Game Theoretic Approach for Addressing Domain-shift with Applications to Big-Data Classification*. This paper is under review with *IEEE Transactions on Knowledge Data and Engineering*. Finally, Paper V covers pages 163-201 and the title is *Distributed Min-max Learning Scheme for Neural Network with Applications to High Dimensional Classification*. This paper is in review with *IEEE Transactions on Neural Networks and Learning Systems*.

## ABSTRACT

In this digital age, big-data sets are commonly found in the field of healthcare, manufacturing and others where sustainable analysis is necessary to create useful information. Big-data sets are often characterized by high-dimensionality and massive sample size. High dimensionality refers to the presence of unwanted dimensions in the data where challenges such as noise, spurious correlation and incidental endogeneity are observed. Massive sample size, on the other hand, introduces the problem of heterogeneity because complex and unstructured data types must analyzed. To mitigate the impact of these challenges while considering the application of classification, a two step analysis approach is introduced where the first step is that of dimension-reduction and the second step is classification using deep neural networks(DNN).

First, multi-step dimension-reduction approaches are developed where both linear and nonlinear relationships among dimensions are considered. The dimensions are first grouped, and each group is then transformed. These two stages are repeated for multiple steps to achieve sufficient dimension-reduction. Novel singular value-based criterions are defined to create groupings, determine the number of steps, and control information loss. Mitigation of noisy dimensions with reduced computational complexity is demonstrated.

Subsequently in the second part of the dissertation, it is shown that heterogeneity can increase generalization error in DNN-based classification. To mitigate this effect, novel learning frameworks are developed where generalization cost is approximated and then minimized during learning. Direct error-driven learning scheme and an additional variables-based distributed learning regime are introduced to train the DNN in the presence of big-data challenges. Efficient DNN learning is demonstrated while mitigating vanishing gradients and challenges due to sparse non-convex optimization. Simulation results using several big-data sets validate theoretical results.

## ACKNOWLEDGMENTS

I would like to sincerely thank my advisor Dr. Jagannathan Sarangapani whose support and guidance made this work possible. I would like to add an extra token of gratitude to Dr. V.A. Samaranayake, for supporting me through the ups and downs. Furthermore, I would like to make a special mention to my mother Mrs Jayalakshmi Raghavan and my father Mr. P.A. Raghavan for being my rock. Additionally, I would like to thank Dr. Donald Wunsch, Dr. Maciej Zawodniok and Dr. Ronald Stanley for serving on my committee. I also thank the NSF and the Intelligent Systems Center for financially supporting the research.

I would like to extend my gratitude to my friends who have supported me through the years. The list includes Dr. Vignesh Narayanan, Dr. Elizabeth Bowles, Mr. Sivaramakrishnan Jayaram, Dr. Maigha Garg, Mr. Siddharth Ashokkumar, Mr. Siddharth Gopalakrishnan, Mr. Anand Nambisan, Mr. Churchil Roy Garg, Mr. Manoj Vishwanathan, Ms. Rajashri Ghorude, Ms. Prashanthi Metku and many more. Without their support, I would be nowhere. Thank you all.

## TABLE OF CONTENTS

## LIST OF ILLUSTRATIONS

# LIST OF TABLES

PAPER V

## SECTION

## 1. INTRODUCTION

Digital data is ubiquitous. Applications such as computer vision, health-care and manufacturing systems generate considerable amounts of data. For instance, a Boeing jumbo jet generates about 650 TB of data over one Atlantic crossing (Figure 1.1). Such data-sets are typically called big data.

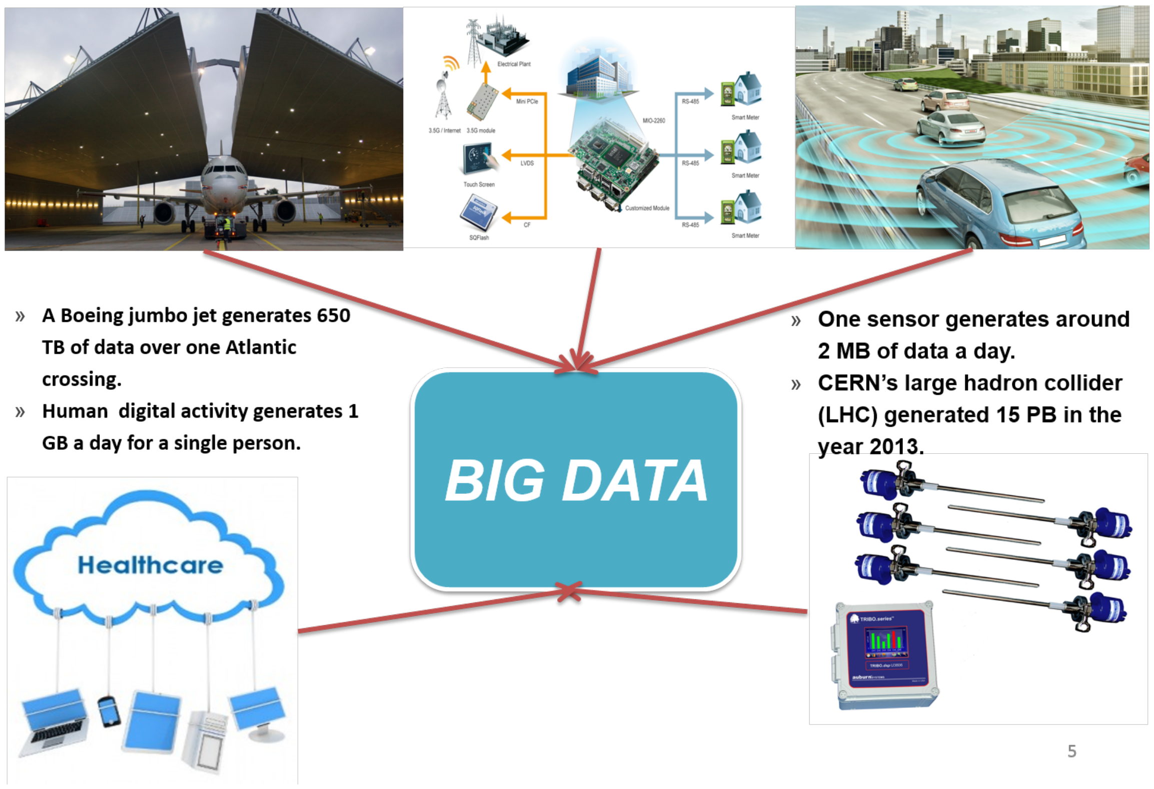

» A Boeing jumbo jet generates 650 TB of data over one Atlantic crossing.
» Human digital activity generates 1 GB a day for a single person.
» One sensor generates around 2 MB of data a day.
» CERN's large hadron collider (LHC) generated 15 PB in the year 2013.

Figure 1.1. Big data generated by various fields.

Big data refers to data-sets whose size is beyond the ability of typical database software tools to capture, store, manage, and analyze. This definition is intentionally subjective. For instance, the advancement in computing technology might change the definition of how big a data-set must be before it can be classified as big data.

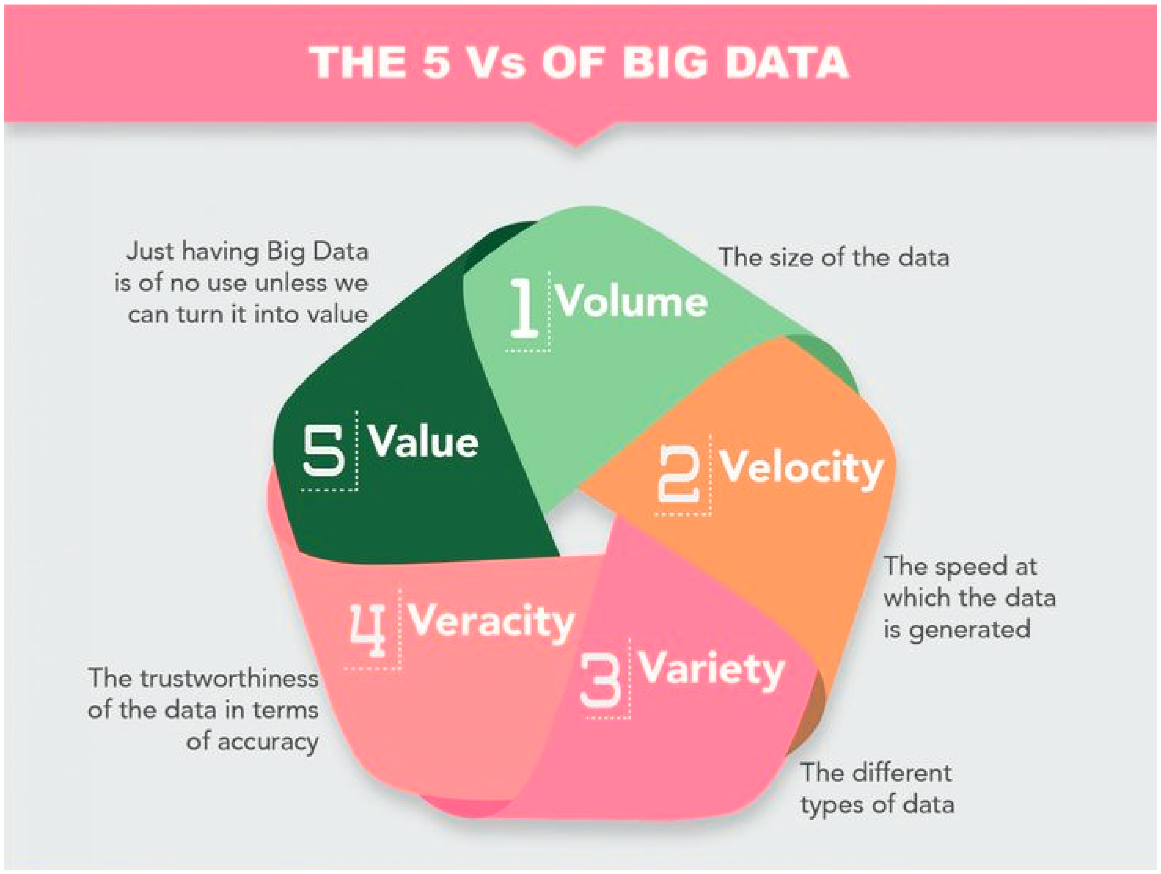

THE 5 Vs OF BIG DATA
1 Volume
The size of the data
2 Velocity
The speed at which the data is generated
3 Variety
The different types of data
4 Veracity
The trustworthiness of the data in terms of accuracy
5 Value
Just having Big Data is of no use unless we can turn it into value

Figure 1.2. Characteristics of big data [1].

The term big data does not capture the quantity of data alone but is also characterized by the 5Vs, which include volume, velocity, veracity, value and variety. Volume refers to the amount of data available whereas velocity refers the fast moving nature of data. Variety indicates that the data is collected from various sources. Veracity represents the uncertainty presented by the data. Finally, value refers to the amount of actionable insight provided by the data . These 5 Vs can be used to loosely define big data. See Figure 1.2 for an illustration.

Through efficient analysis, a multitude of important information can be extracted from big data to drive key decision making. Because the 5Vs of big data distinguish it from traditional data sources, a dramatic paradigm shift must occur in analysis strategies. Such a paradigm shift highlights the fact that strategies required for sustainable analysis in big data scenarios are fundamentally different from traditional methodologies [2].

Analysis strategies in big data can be categorized into techniques and technologies. Technologies refer to software solutions that focus on developing infrastructures to store, retrieve and enable analysis of big data [3]. These methodologies [3] are primarily focused on developing computationally friendly solutions to the problem of analyzing big data. In this process, the data is usually broken down into smaller manageable chunks. A distributed processing environment such as Hadoop or Apache spark [5, 6] is then used to analyze them. The analysis of these independent sections of data is usually performed with traditional data analysis methods [3] such as principal component analysis (PCA), K-nearest neighbors, etc. This field is not the focus of this dissertation.

On the other hand, the techniques required for the sustainable extraction of actionable insights from big data are dedicated to understanding and addressing the unique statistical challenges introduced by large dimensionality and the volume of big data. Typically, the challenges to be addressed depend on the application in hand. In this dissertation, we consider the problem of classification where the objective is to determine which category does a data-point belongs to. To predict these categories, the main idea is to estimate a map $\phi(.)$, referred to as classifier, that transforms the data-point $\boldsymbol{x}$ into a probability space where the magnitude of $\phi(\boldsymbol{x})$ indicates the membership of $\boldsymbol{x}$ to different categories. Membership values signify the probability of $\boldsymbol{x}$ belonging to any particular category. To learn $\phi(.)$ one usually assumes that a data-set $\boldsymbol{\mathcal{X}}$ with labels $\boldsymbol{\mathcal{Y}}$ is available and it represents all the necessary information. The overall learning then involves approximating $\phi(.)$ through a parametric map such as neural networks, spline functions, etc.

While classifying high dimensional data-sets, challenges such as noise accumulation, spurious correlation and incidental endogeneity can be observed [3, 2]. In addition, large volume of data also introduces heterogeneity and statistical bias. High dimensionality combined with large volume introduces high computational cost.

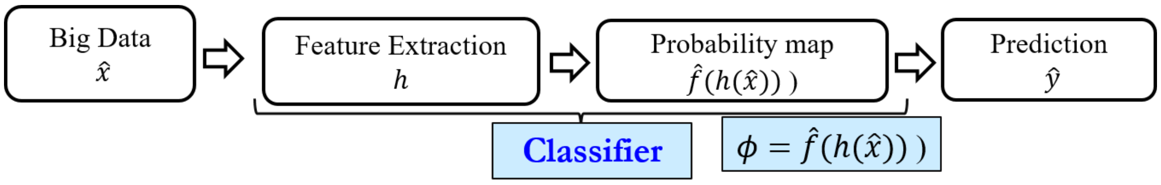


Figure 1.3. Big data classification pipeline.

The problem of classification can be solved using two step. Each step in the process is a map as described in Figure 1.3 such that feature extraction map is denoted by $\hat{f}$ and probability map is denoted by $\hat{h}$. The composition of the two maps is referred to as a "classifier." Specifically, feature extraction involves obtaining a subset of features from the data-set, where this subset is intended to be informative and non-redundant. These extracted features can subsequently facilitate the efficient design of the probability map [4]. The probability map can involve a variety of applications such as fault diagnostics and classification [3]. In a traditional setup, one would design a feature extraction map and a probability map independently whereas with deep neural network-based designs, one usually combines the two steps into a single map and learn it efficiently through optimization approaches. Typically, challenges such as noise, spurious correlations and heterogeneity are observed in the feature extraction process whereas the problem of incidental endogeneity and heterogeneity arises in the probability map.

Noise can be of two types: noisy dimensions - dimensions in the data that do not contribute to analysis, and noisy data - the presence of distortions in the data. The accumulation of noisy dimensions and noisy data can result in severe degradation of performance of the prediction model [3].

Spurious correlation refers to the case, when two dimensions exhibit high values of sample correlations but there is no causal relationships between these dimensions. Critical problems may arise with spurious correlation in scenarios where the number of dimensions far exceeds the number of data-points [2]. In such cases, it has been shown [3, 7] that only a few dimensions contribute positively to analysis whereas the other dimensions introduce errors and contribute to spurious correlation [7].

Big data-sets are often created by aggregating different data sources. Each data source can exhibit unique characteristics not shared by other sources which results in the problem of heterogeneity. Remarkable insights can be derived if unique characteristics from different sources are considered during analysis. Another unique challenge introduced in high dimensional regression problems is incidental endogeneity which is the scenario when the regression parameters are directly correlated to the residual errors. Incidental endogeneity causes inconsistency in model selection [2, 8].

In spite of these complications, sustainable classification is possible if, careful attention is paid to these challenges. This becomes the motivation of this work. An overview of current literature is discussed next.

## 1.1. OVERVIEW OF CURRENT BIG DATA ANALYSIS METHODOLOGIES

A common solution to the issue of noise accumulation and spurious correlations is to extract features using dimension-reduction methodologies; see Figure 1.3 for an illustration. In the current literature, PCA [11], factor analysis [12], etc. are commonly used for dimension-reduction. In these methodologies [11, 12], a linear projection of the data is performed using eigen-decomposition of the correlation matrices. However, efficient projection requires accurate estimation of dependence relationships.

Two problems that are observed when evaluating dependence relationships in big data. First, when Pearson correlation/covariance matrices are used, imperfect estimation of dependencies is common due to the presence of nonlinear relationships, [7]. Second, when the number of dimensions in the data far exceeds the number of sample points, a non-singular Pearson correlation/covariance matrix cannot be guaranteed [7, 11, 12]. Due to insufficient information about dependencies, efficient feature extraction cannot be ensured.

In the literature, several dimension-reduction techniques such as isomap [13], locally linear embedding (LLE) [14], Hessian LLE [15], Laplacian eigenmaps [16] and kernel PCA [17] have been introduced for addressing nonlinear relationships. These methodologies [13, 14, 15, 16, 17] ascertain the intrinsic geometric structure of high dimensional data. However, high dimensional spaces are sparse and suffer from distance concentration [7] which makes the discovery of intrinsic geometric structure challenging.

Recently divide-and-conquer mechanisms for dimension-reduction were proposed in [18, 19, 20, 21] that use known application-specific relationships and natural groupings to perform group-wise organizaton. Prior information about natural grouping structure is required to enable such organization in these approaches [18, 19, 20, 21] which is not always available in big data scenarios.

One step mapping from a higher to a lower dimensional space is the underlying principle in the approaches mentioned earlier [11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21]. Such approaches [11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21] are computationally unfriendly and susceptible to improper estimation because of noise and redundancies. A multi-step dimension-reduction approach is therefore desirable in big data scenarios [2]. This concept forms one of the fundamental ideas for the first two papers in this dissertation.

A small subset of features is traditionally created in the previously mentioned approaches [11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21] . These subsets can also be created by generate-and-test methods [22, 24]. The main drawback for these methods [11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 24] stems from the fact that the performance of the detection methodology is not taken into account during feature extraction. It has been shown in [30] that the inclusion of such a feedback would assist the feature extraction process, especially in complicated data-sets, where traditional dimension-reduction approaches do not perform well.

Deep neural networks (deep NN) have gained prominence recently for learning-based feature extraction [25] in classification problems. In a typical deep NN, feature extraction is performed using compositions of parametric maps that involve linear and non-linear transformations. A performance measure is generally minimized to learn the mapping parameters [26]. The inclusion of the performance measure allows the feature extractors to receive feedback on their performance from the detection methodology. The procedure involves evaluating the gradient of the performance measure with respect to the parameters, which forms the premise for training deep NN, popularly known as backpropagation [27]. However, the presence of big data-challenges introduces a shift in the data-distribution of the data under analysis. This shift can be observed due to a number of reasons such as data-noise and statistical heterogeneity [2] and it introduces errors, known as generalization errors, in the data analytics including prediction. Furthermore, additional challenges such as incidental endogeneity and vanishing gradients [28] are also observed.

Authors in [30] addressed the issue of generalization error by proposing an ensemble of classifiers. In contrast, the authors in [31] introduced an additional adaptation layer to explicitly model faulty labels. A bounded term in the cost function was proposed in [32] to compensate for the impact of noisy dimensions. In applications involving big-data, the method in [30] is not feasible due to the amount of resources required. Likewise, due to the large number of noisy dimensions in big data, the bounds introduced by [32] may not exist. Moreover, these approaches [32, 30, 31] ignore the effects of noise and only focus on faulty labels. In the literature, regularization techniques such as $L_2$ norm, dropouts [33] and adversarial regularization [34] are used to improve generalization error. Yet, these methods[33, 34] are not systematic. Furthermore, several DNN based approaches have been proposed to compensate for such shift in the data-distribution [40, 41]. These approaches suffer from critical drawbacks in big data environments which can be summarized as follows: (1) the performance depends on the good use of heuristics [37, 38, 39], which is difficult to obtain in the big data environments: and (2) they work well under a general assumption: the distribution shift is exactly known [41, 40], an impractical assumption in many engineering environments.

The issue of vanishing gradients are typically mitigated in the literature using alternate activation functions such as leaky relu, drelu, etc [43, 28]; or learning rate schedules [28] are common ways of addressing this issue. However, it has been shown that these approaches [28] do not mitigate the vanishing gradient problem completely and only provide a workaround for satisfactory performance [43].

These drawbacks have motivated further research in alternate non-gradient-based learning frameworks [46, 47]. Target propagation was introduced in [46] where difficult to define targets are introduced at each layer with great computational cost to break the learning process across layers. In addition, error-driven learning approaches were proposed in [47] which suffer from inefficiency while optimizing convolutional neural networks (CNN) as is shown in [47]. The presence of high dimensional data-sets also introduces additional challenges into the learning procedure [2]. As the number of parameters in the DNN scales quickly with dimensionality of data estimating parameters of DNN mapping is intractable because enough data-points are not available. Additionally challenges such as spurious correlation and incidental endogeneity introduces estimation errors into the learning problem.

All of these challenges reflect the need for mitigating the impact of big data challenges on the traditional data-analysis methodologies and thus motivated by these facts, a suite of methodologies are presented in this dissertation for the analysis of big data. The techniques proposed in this dissertation can be broadly categorized into dimension-reduction-based approaches and learning-based methodologies. The organization of this dissertation is presented next.

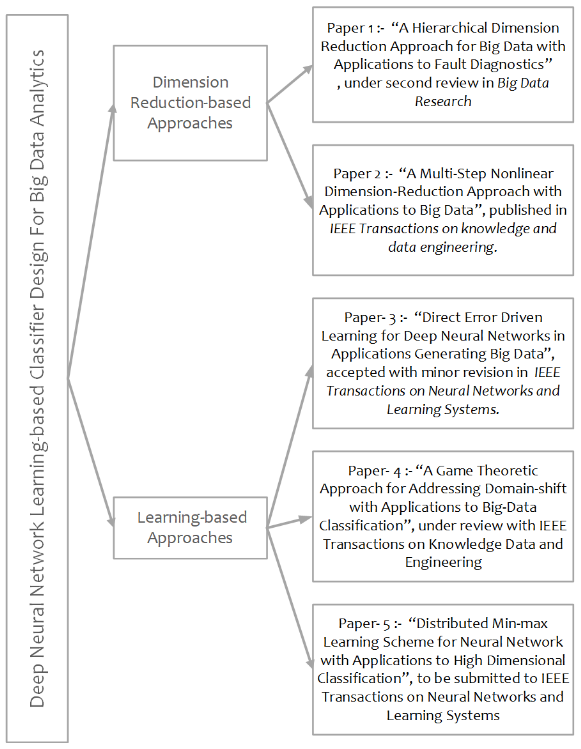

Deep Neural Network Learning-based Classifier Design For Big Data Analytics
Dimension Reduction-based Approaches
Learning-based Approaches
Paper 1 :- "A Hierarchical Dimension Reduction Approach for Big Data with Applications to Fault Diagnostics" , under second review in *Big Data Research*
Paper 2 :- "A Multi-Step Nonlinear Dimension-Reduction Approach with Applications to Big Data", published in *IEEE Transactions on knowledge and data engineering.*
Paper- 3 :- "Direct Error Driven Learning for Deep Neural Networks in Applications Generating Big Data", accepted with minor revision in *IEEE Transactions on Neural Networks and Learning Systems.*
Paper- 4 :- "A Game Theoretic Approach for Addressing Domain-shift with Applications to Big-Data Classification", under review with IEEE Transactions on Knowledge Data and Engineering
Paper- 5 :- "Distributed Min-max Learning Scheme for Neural Network with Applications to High Dimensional Classification", to be submitted to IEEE Transactions on Neural Networks and Learning Systems

Figure 1.4. Organization of the dissertation.

## 1.2. ORGANIZATION

This dissertation will be covered in five papers. The relation of one paper to another is illustrated in Figures. 1.4 and 1.5. As a common theme, the problem of classification is considered in this dissertation. In the first two papers, the focus is on the design of the feature extraction map whereas deep NN-based approaches are presented in the last three papers.

In the first paper, a hierarchical dimension-reduction approach is presented for distance measure based faults diagnostics in manufacturing systems. This approach assumes that linear relationships are present in the data. Specifically, a generalized distance measure is proposed in conjunction with a novel hierarchical dimension-reduction (HDR) approach is presented. Redundant dimensions and non-invertible correlation matrices are addressed.

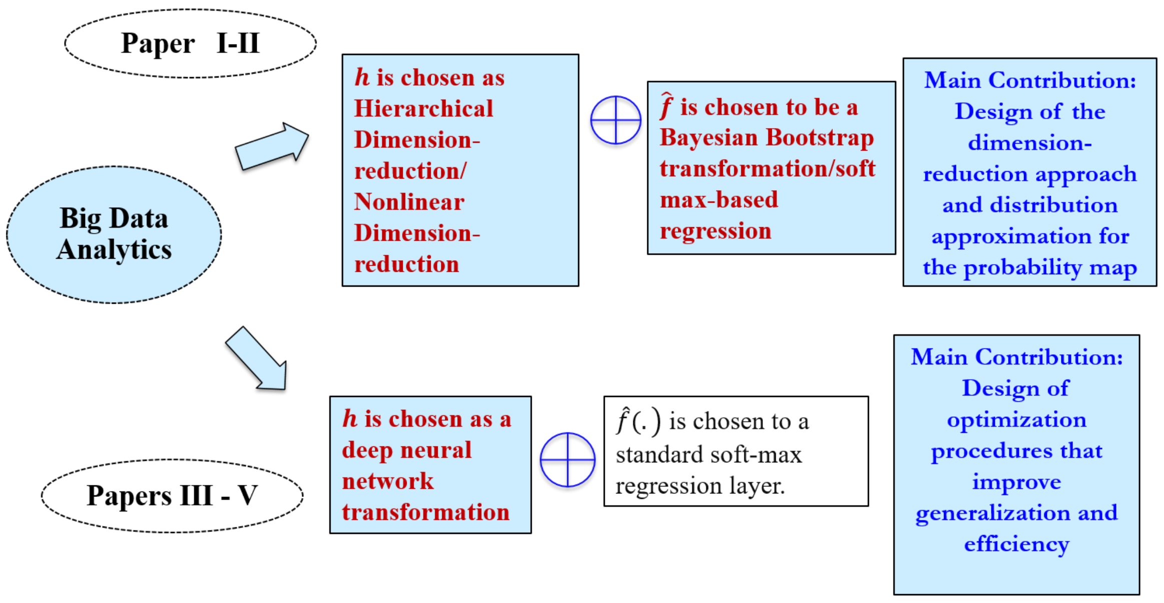

Paper I-II
Big Data Analytics
h is chosen as Hierarchical Dimension-reduction/ Nonlinear Dimension-reduction
$\hat{f}$ is chosen to be a Bayesian Bootstrap transformation/soft max-based regression
Main Contribution: Design of the dimension-reduction approach and distribution approximation for the probability map
Papers III - V
h is chosen as a deep neural network transformation
$\hat{f}(.)$ is chosen to a standard soft-max regression layer.
Main Contribution: Design of optimization procedures that improve generalization and efficiency

Figure 1.5. Organization of the dissertation with reference to the big data analysis pipeline.

The task of isolation and detection of faults is performed using a probabilistic methodology developed in this paper. To approximate the density function of the data in the presence of high dimensions, Edgeworth expansion-based expressions are derived.

In the second paper, the assumption of linearity from the first paper is relaxed. Challenges such as heterogeneity and noisy dimensions are addressed in the context of dimension-reduction. In the proposed approach, dimensions in the data are reduced while considering the data in batches. Nonlinear relationships are captured using distance correlation matrices. Redundant and noisy dimensions are systematically removed by dividing the attributes in the data into groups and using a parametric map in each group for transformation into a low dimensional space. The magnitude of singular values of the respective attributes is used to enable group-wise organization. The organization and subsequent reduction process is performed until a user-defined criterion on singular values is satisfied. The problem of heterogeneity is addressed by introducing matrix factorization-based aggregation methods. Data in the reduced dimensions are utilized for fault diagnostics and classification. Due to the lack of any feedback from the detection model, NDR and HDR underperformed in complicated data-sets such as CiFar-10.

To enable this feedback, and to minimize the impact of noisy data, heterogeneity while learning to classify data points, direct error-driven learning scheme is proposed in the third paper. First, to reduce the impact of heterogeneity and data-noise, the concept of a neighborhood is introduced. The neighborhood is generated by introducing perturbations into the training data and generating additional samples. Using these additional samples, an approximation of generalization error is obtained and an overall error, comprised of learning and the approximate generalization errors, is defined. A novel NN weight-tuning law is obtained through a layer-wise performance measure that enables the direct use of overall error for learning. Through a comprehensive simulation study, the proposed approach is shown to mitigate the vanishing gradient problem while improving generalization by 6%.

In the previous paper, the extent of perturbations that must be introduced to generate an approximation of generalization error was chosen by the practitioner. To mitigate this issue, in the fourth paper, we introduce a distortion model that introduces these perturbations(domain shift) into the training data. Then, through a two player zero sum game that involves a deepNN-based classifier, we solve for optimal shift and its compensation through the classifier. In the proposed game, the distortion model is viewed as the maximizing player which increases the domain shift while the classifier becomes the minimizing player that improves classification accuracy. The Nash solution of the game provides the domain shift and its optimal adaptation through the classifier. To solve the proposed game for the Nash solution, the direct error-driven learning scheme that was introduced in the last paper is modified. A comprehensive simulation study and mathematical analysis is presented to demonstrate the efficiency.

In the last paper of the dissertation, an alternative learning methodology is introduced to address challenges due to high dimensional data-sets. Here, a novel distributed learning methodology is introduced for deep NN while considering a $L_1$ regularized optimization. Here, optimal sparsity in the deep NN is estimated by designing a two player zero-sum game between the penalty coefficients/sparsity parameters and the DNN weights. To solve this game, a novel learning scheme is introduced where layer wise cost functions are derived at each DNN layer by introducing additional variables. As a result, the output at a given DNN layer depends on the additional variables instead of the previous layer. An alternating-minimization optimization procedure is designed to optimize these layer-wise cost functions. The obtained Nash solution of the game provides optimal sparsity and its compensation through the classifier. The proposed approach is finally implemented in a parallel and distributed environment through a novel computational algorithm and demonstrated both theoretically and using nine data-sets.

## 1.3. CONTRIBUTIONS

The techniques presented in this dissertation can be broadly categorized into dimension-reduction-based approaches as presented in Papers 1-2 and learning-based approaches are presented in Papers 3-5. The contributions for Paper 1 include the: (1) development of a revised distance measure based on a novel hierarchical dimension-reduction approach (GDM-HDR) for efficiently calculating distances in large dimensional cases; (2) investigation of GDM-HDR-based statistical test procedures for fault detection; (3) introduction of a generalized distance-based classification scheme for big data diagnostics where bootstrap sampling is used and Edgeworth expansion-based expressions are derived to approximate the density function for each fault; and (4) demonstration of the analytical efficiency with the computational benefits of the GDM-HDR-based approach in a big data scenario.

Paper 2 alleviates the assumption of linearity from Paper 1. The contributions include the (1) development of an multi-step, group-wise dimension-reduction approach using distance covariance (NDR) to handle nonlinear relationships and noisy dimensions; (2) design of a matrix factorization-based mechanism to update distance covariance and the corresponding SVD using the data in batches to address heterogeneity; (3) introduction of a generic group-wise organization procedure with explicit stopping criterion to control information loss; and (4) demonstration of the efficacy of NDR using nine data-sets covering the problem of classification and fault diagnostics.

Paper 3 introduces a learning-based framework for analyzing big data. The contributions include (1) development of a novel cost formulation for optimization of deep NN-based classification to reduce the impact of heterogeneity and data-noise; (2) design of a direct error-driven learning scheme is defined to minimize the cost function to address heterogeneity and data-noise while mitigating the vanishing gradient problem; (3) development of a systematic approach to mitigate the impact of noisy dimensions to learning for the EDL scheme and (4) demonstration of the efficacy of the proposed approach with simulation analysis.

Paper 4 addresses the problem of unknown neighborhood size where the contributions are (1) generic distortion model for DA in the presence of big data challenges such as heterogeneity and noise; (2) design of a two player zero-sum game for classification; (3) direct error driven alternating minimization approach for solving the zero-sum game; (4) simulation and mathematical analysis of the proposed approach involving a total of ten data-sets.

Finally, Paper 5 presents a distributed learning approach. (1) novel distributed learning approach of DNN is proposed. (2) a two player game is developed for mitigating the impact of high dimensionality of data in the model. (2) theoretical results demonstrating the validity of the proposed approach. (3) efficiency of the approach is demonstrated with simulation results involving the use of benchmarking data-sets. (4) development of a parallelized implementation algorithm for the proposed approach. Each paper in the dissertation is individually presented next.

## PAPER

# I. A HIERARCHICAL DIMENSION REDUCTION APPROACH FOR BIG DATA WITH APPLICATION TO FAULT DIAGNOSTICS

R. Krishnan

Department of Electrical & Computer Engineering

Missouri University of Science and Technology

Rolla, Missouri 65409–0050

Email: krm9c@mst.edu

V.A. Samaranayake

Department of Mathematics & Statistics

Missouri University of Science and Technology

Rolla, Missouri 65409–0050

Email: vsam@mst.edu

S. Jagannathan

Department of Electrical & Computer Engineering

Missouri University of Science and Technology

Rolla, Missouri 65409–0050

Email: sarangap@mst.edu

## ABSTRACT

About four zetta bytes of data, which falls into the category of big data, is generated by complex manufacturing systems annually. Big data can be utilized to improve the efficiency of an aging manufacturing system, provided, several challenges are handled. In this paper, a novel methodology is presented to detect faults in manufacturing systems while overcoming some of these challenges. Specifically, a generalized distance measure is proposed in conjunction with a novel hierarchical dimension-reduction (HDR) approach. It is shown that the HDR can tackle challenges that are frequently observed during distance calculation in big data scenarios, such as norm concentration, redundant dimensions, and a non-invertible correlation matrices. Subsequently, a probabilistic methodology is developed for isolation and detection of faults. Here, Edgeworth expansion based expressions are derived to approximate the density function of the data. The performance of the dimension-reduction methodology is demonstrated to be efficient both theoretically and empirically with simulation results involving the use of big data sets. It is shown that HDR is able to explain almost 90% of the total information. Furthermore, the proposed dimension-reduction methodology is seen to outperform standard dimension-reduction approaches and is able to improve the performance of standard classification methodologies in high dimensional scenarios.

## 1. INTRODUCTION

With the explosion of digital data, a dramatic paradigm shift has taken place in the field of data analysis. Such a dramatic shift has highlighted the fact that strategies required for sustainable analysis in big data scenarios are fundamentally different from traditional methodologies [1].

Big data is characterized by unwanted, incomplete and redundant dimensions [2, 3] where complications such as spurious correlations and incidental endogeneity may be observed. Spurious correlation is seen when completely uncorrelated dimensions indicate sample correlation [1], whereas incidental endogeneity is the presence of a relationship between unwanted dimensions and the predictor. These complications lead to false estimation of dependencies in the data that result in erroneous fault detection.

When high dimensionality is mixed with massive data, there is a need for distributed storage solutions [4]. As a consequence, one must rely on samples for analysis, necessitating the effective aggregation of parameters. In the absence of such a process, several complications, including heterogeneity, experimental variation and statistical bias can be seen [1].

In spite of the serious complications discussed earlier, sustainable analysis in several fields, including statistics [5], machine learning [5], security [5] and fault diagnostics [6, 7] is possible if serious attention is paid to the aforementioned challenges. The remedies to the above problems are discussed through the setting of a fault diagnostics application.

There are typically two aspects to fault diagnostics. The first is fault detection where an abnormal behavior is observed in an aging manufacturing system. The second is identification of the fault type that is in progress. A logical way to address these problems is to use a distance measure to analyze the deviation from healthy case while using a classification methodology for recognizing them.

Several approaches have been proposed in the literature [8, 6, 7], that incorporate a similarity measure for the problem of diagnostics. In these efforts, Mahalanobis distance (MD) [9] has been demonstrated as a capable distance measure for multi-dimensional scenarios.

However, in a big data scenario, two problems arise. First, when the sample size is close to the number of dimensions or when there are redundant, unwanted dimensions in the data, the covariance/correlation matrix becomes singular or non-invertible [2, 3]. As a consequence, correlation matrix cannot provide enough information and distance calculation becomes meaningless. Second, distance values concentrate and lose discriminative power in high dimensional spaces [10, 3, 11].

A popular solution to address these issues is to reduce the number of dimensions prior to distance calculation. Recently, several researchers [6, 12, 8, 7] have introduced such techniques for fault diagnostics. In these efforts, popular dimension reduction methods such as principal component analysis (PCA) [6, 13], factor analysis are frequently used for reducing dimensions.

However, these methods [6, 12, 8, 7] rely on correlation or covariance matrices for dimension reduction. Due to the challenges posed by high dimensional spaces discussed before, it is not possible to estimate these matrices correctly. Moreover, the process is computationally infeasible.

To address the problem of computational disadvantages and imperfect estimation in large dimensional scenarios, divide-and-conquer mechanisms for dimension reduction were proposed in [15, 16, 17, 18], where the authors have shown that in many cases, known application-specific relationships and natural groupings can be exploited for efficient dimension reduction. However, such approaches [15, 16, 17, 18], require prior knowledge of these relationships to enable organization, which may not be available in big data scenarios.

Furthermore, in all of these dimension reduction approaches [6, 12, 8, 7, 19, 14, 13, 15, 16, 17, 18], the underlying principle is to perform one-step mapping from a higher dimensional space to a lower dimensional space. First, such an approach is computationally unfriendly when the dimension of the original data is very large. Moreover, it is also susceptible to improper estimation due to the presence of large amounts of noise and redundancies [29]. In light of these complications, a generalized distance measure based on a novel hierarchical dimension reduction approach (GDM-HDR) is presented in this paper.

In the proposed split and merge approach, dimensions in the data are reduced prior to distance calculation. Specifically, the dimensions in the data are organized into groups and dimension are reduced in each group using the eigen-decomposition of the correlation matrix.

First, by grouping the dimensions and by controlling the size of the group, One can avoid the estimation of a very large correlation matrix and at the same time fulfill the rank condition for correlation/ covariance matrices. Second, systematic removal of unwanted dimensions is performed in the proposed methodology. Third, a step-by-step process of evaluation enables a distributed estimation of dependence relationships thus enabling parallelization. Lastly, using a fractional-norm to induce distance instead of the classical $L_2$ norm reduces concentration of distances. As a side benefit, the overall computational overhead of the calculation is linear with the number of dimensions in the data, making the methodology favorable in large scale applications.

Similar to others [14, 12, 6, 7], a divide and conquer strategy is incorporated in this paper. However, this work is fundamentally different from that in [14, 12, 6, 7]. Here, the problem is solved by dividing the data across dimensions rather than along data-points, which results in important advantages in big data scenarios. In contrast with [6, 12, 8, 7, 19, 14, 13, 15, 16, 17, 18], a multi-step dimension-reduction approach is introduced in HDR to reduce inconsistencies and redundancies. It is shown that the complexity of dimension reduction is reduced from a quadratic complexity to linear complexity.

To test the efficiency of the proposed distance measure, a comprehensive methodology is developed for diagnostics. Here, bootstrap sampling-based statistical tests are utilized for detecting faults. Further, a bootstrap distribution based Bayesian methodology is proposed to isolate faults. Here, Edgeworth expansion based approximations are derived for distribution estimation, which allows for uncertainty modeling even for non-Gaussian distributions.

The overall methodology is shown to be very effective for diagnostics in complex manufacturing systems generating big data with thorough theoretical and simulation analysis. Although HDR and the proposed fault isolation methodology are designed specifically for the application of fault diagnostics, they are applicable to a larger class of problems such as classification. A comprehensive study is therefore presented to compare the pro-

Table 1. Notations

| **Notation** | **Meaning** |
|---|---|
| $\boldsymbol{\mathcal{X}}_0, \boldsymbol{\mathcal{Y}}$ | Healthy sample, Test sample |
| $\mathcal{F}$ | Number of faults in the system. |
| $\boldsymbol{\mathcal{X}}_k : \quad k = 1, 2, \cdots, \mathcal{F}$ | Samples corresponding to the faults. |
| $n$ | Number of attributes in the data. |
| $\mathcal{M}_0, \quad \mathcal{M}_y, \quad \mathcal{M} = \cup_{k=0}^{\mathcal{F}} \mathcal{M}_k$ | Number of samples in the healthy sample, number of samples in the test sample, Total number of samples. |
| $p : \quad p = 1, 2, \cdots \mathcal{M}_0 \quad p' : p' = 1, 2, \cdots \mathcal{M}_y$ | Index for the healthy sample and the test sample. |
| $(i) : i = 1, 2, \cdots I$ | Level index in the dimension reduction process. |
| $t : \quad 1, 2, 3, \cdots T^{(i)}$ | Group index at any particular level $(i)$. |
| $r : \quad r = 1, 2, 3, \cdots \eta_t^{(i)}$ | Node index at group $t$ and level $(i)$. |
| $\boldsymbol{C}_t^{(i)}$ | Correlation matrix at group $t$ and level $(i)$. |
| $\boldsymbol{e}_t^{(i)}$ | Eigenvector at group $t$ and level $(i)$. |
| $\boldsymbol{\mathcal{H}}_t^{(i)}$ | Standardization matrix at group $t$ and level $(i)$. |
| $Q(c\|\boldsymbol{y})$ | Posterior probability that a data-point $y$ belongs to class $c : \quad c = 1, 2, \cdots, \mathcal{F}$. |
| $\pi_c$ | Prior probability that a data-point $y$ belongs to class$c : \quad c = 1, 2, \cdots, \mathcal{W}$. |
| $\delta(\boldsymbol{\mathcal{X}}_0, \boldsymbol{\mathcal{Y}})$ | GDM-HDR between two samples. |

posed methodology with other dimension reduction approaches such as PCA, FA, KPCA, etc. Furthermore, the performance of HDR in conjunction with standard classification methodologies is also studied here. From the analysis the proposed methodology is seen to outperform standard dimension reduction approaches in high dimensional cases.

The contributions of this paper therefore include (1.) the development of a revised distance measure based on a novel hierarchical dimension reduction approach (GDM-HDR) for big data, (2.) the development of GDM-HDR-based statistical test procedures for fault detection, (3.) the introduction of a generalized distance-based classification scheme for big data diagnostics, (4.) mathematical analysis for GDM-HDR and (5.) demonstration of the analytical efficiency along with the computational benefits of GDM-HDR-based approach in a big data scenario.

The rest of the paper is organized as follows. Section II describes Mahalanobis distance and establishes motivations for the paper. Section III describes the proposed approach. Section IV outlines the fault diagnostics methodology. Sections V and VI detail the simulation results. Section VII provides conclusions for the paper. Required notations and preliminaries are discussed in the following section.

## 2. NOTATIONS AND PRELIMINARIES

To establish notation, let $\mathbb{R}$ be the set of real numbers and $\mathbb{N}$ the set of natural numbers. Matrices and vectors are represented in bold by uppercase and lowercase symbols respectively. Data is sampled from one of many populations gathered from $n$ attributes. All the notations are summarized in Table 1

Let $\boldsymbol{\mathcal{X}}_0$ contain a random sample of size $\mathcal{M}_0$ obtained from a population of $n$-variate random vectors with mean and covariance given as $\boldsymbol{\mu}_0$ and $\boldsymbol{\Sigma}_0$ respectively. Without the loss of generality, $\boldsymbol{\mathcal{X}}_0 = \{\boldsymbol{x}_p : p = 1, 2, 3, 4, \cdots, \mathcal{M}_0, \boldsymbol{x}_p \in \mathbb{R}^n\}$, where $\mathcal{M}_0 \in \mathbb{N}$. Let $\boldsymbol{\mathcal{X}}_k$ contain a random sample of size $\mathcal{M}_k$ from a $n$-variate random populations. Further let $\mathcal{M} = \cup_{k=0}^{\mathcal{F}} \mathcal{M}_k$ be the total number of sample points in the dataset.

Sample mean vector and covariance matrix for any population

$$k : k = 0, 1, 2, 3, \cdots, \mathcal{F}$$

, are denoted as $\overline{\boldsymbol{x}}_k$ and $\hat{\boldsymbol{\Sigma}}_k$ respectively. Data vectors in $\boldsymbol{\mathcal{X}}_k$ are assumed to be collected when fault $k$ is introduced into the manufacturing system intentionally. Consider $\boldsymbol{\mathcal{X}}_0$ to contain data vectors collected when the system is healthy. Vectors in $\boldsymbol{\mathcal{X}}_k$ are assumed to be independent of $\boldsymbol{\mathcal{X}}_{k^*}$ for $k \neq k^*, k, k^* = 0, 1, 2, \cdots \mathcal{F}$. The following assumption is considered:

**Assumption 1** *Whenever the data sample $\boldsymbol{\mathcal{X}}_k, k = 0, 1, 2, 3, \cdots, \mathcal{F}$ is collected, only one fault is considered present.*

Let $\boldsymbol{\mathcal{Y}} = \{\boldsymbol{y}_p : p = 1,2,3,4,\cdots,\mathcal{M}_y, \boldsymbol{y}_p \in \mathbb{R}^n\}$ be another random sample of data, obtained when the system is operating freely without intervention but under one of the possible "fault" conditions $k = 0,1,2,3,\cdots,\mathcal{F}$. The data vectors in $\boldsymbol{\mathcal{Y}}$ can be seen to constitute a random sample with the sample mean and covariance given respectively as $\overline{\boldsymbol{y}}$ and $\hat{\boldsymbol{\Sigma}}$.

**Remark 1** *Note that data vectors in $\boldsymbol{\mathcal{Y}}$ are observed when the system is either operating under the healthy condition or when the system is encountering exactly one of the $\mathcal{F}$ faults.*

It is of practical importance is to determine whether the observations in $\boldsymbol{\mathcal{Y}}$ are obtained under the healthy scenario ($k = 0$) or under one of the faults $k$ such that $k = 1,2,3,\cdots,\mathcal{F}$. One way to accomplish this is by observing the difference between the sample mean $\overline{\boldsymbol{y}}$, obtained from the observations in $\boldsymbol{\mathcal{Y}}$, and the mean $\overline{\boldsymbol{x}}_0$ of the sample, obtained when the system is under the healthy condition. An example of a traditional squared measure for observing this deviation is discussed next.

In [30], the MD between $\overline{\boldsymbol{y}}$ and $\overline{\boldsymbol{x}}_0$ is defined as

$$d(\overline{\boldsymbol{y}}, \overline{\boldsymbol{x}}_0) = (\overline{\boldsymbol{y}} - \overline{\boldsymbol{x}}_0)^T \boldsymbol{\Sigma}^{-1} (\overline{\boldsymbol{y}} - \overline{\boldsymbol{x}}_0). \tag{1}$$

It follows that

$$d(\overline{\boldsymbol{y}}, \overline{\boldsymbol{x}}_0) = \|\boldsymbol{\Sigma}^{-1/2} (\overline{\boldsymbol{y}} - \overline{\boldsymbol{x}}_0)\|. \tag{2}$$

where $\boldsymbol{\Sigma}^{-1/2}$ denotes the Cholesky decomposition [13] of the covariance matrix and $\|.\|$ is the 2-norm. In practice, $\boldsymbol{\Sigma}$ is replaced by its sample estimate $\hat{\boldsymbol{\Sigma}}$.

**Remark 2** *Note that the attributes are standardized prior to distance calculation, as a result, the estimate of covariance is equal to that of correlation for the sample. In other words, $\hat{\boldsymbol{\Sigma}}$ in (2) can be replaced by $\hat{\boldsymbol{R}}$ such that $\hat{\boldsymbol{R}}$ is the estimate of the theoretical correlation matrix corresponding to the covariance matrix $\boldsymbol{\Sigma}$. One can write*

$\boldsymbol{R} = [diag(\boldsymbol{\Sigma})]^{-1/2} \quad \boldsymbol{\Sigma} \quad [diag(\boldsymbol{\Sigma})]^{-1/2}$ *, where* $diag(\boldsymbol{\Sigma})$ *is a diagonal matrix, whose diagonal elements are that of* $\hat{\boldsymbol{\Sigma}}$*. Similarly,* $\hat{\boldsymbol{R}} = [diag(\hat{\boldsymbol{\Sigma}})]^{-1/2} \quad \hat{\boldsymbol{\Sigma}} \quad [diag(\hat{\boldsymbol{\Sigma}})]^{-1/2}$*. From this point forward,* $\hat{\boldsymbol{R}}$ *will be denoted by* $\boldsymbol{C}$ *for simplicity.*

Encountering large dimensions in big-data applications is a major issue faced by data analysts. To mitigate this challenge, in the proposed approach, the vectors $\bar{\boldsymbol{x}}_0$ and $\bar{\boldsymbol{y}}$ are transformed into smaller dimensions. The process is structured in such a way that there are only two steps. In the first step, the attributes are split into groups, facilitating an efficient dependency estimation. Secondly, a merge process is then used as a second step to select projections of attributes that maximize group-wise variability. This is done in an iterative fashion to ensure all dependencies is taken into account.

By performing the above steps, one can extract a revised set of attributes from the data that are linear combinations of the original dimensions. Furthermore, the process ensures that redundant dimensions are removed where they will not impact the distance calculation. Once the vectors are transformed into, say $\bar{\boldsymbol{x}}_0{}^*$ and $\bar{\boldsymbol{y}}^*$, a $L_p$ norm is used to induce the final distance value. The proposed distance measure is explained with all details in the following section.

## 3. GENERALIZED DISTANCE MEASURE: A HIERARCHICAL DIMENSION REDUCTION APPROACH

For a simple illustration of the proposed procedure, consider the case where each data-point is $p-$dimensional. Given a data-point $\boldsymbol{x}$, let these $p$ attributes be denoted by $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_p$. In the context of fault diagnosis, assume that the training data belongs to a healthy scenario with sample $\boldsymbol{\mathcal{X}}$ and the results of the fault diagnostics will be evaluated on a test sample that is $\boldsymbol{\mathcal{Y}}$.

The proposed HDR method would build a fault diagnosis tool or a generic classification tool through a series of steps. In each step, the dimensions of each data points are reduced comprising of two stages: a grouping stage where the attributes are grouped, and a transformation stage where attributes in each group are linearly transformed through principal component analysis (PCA). In this process, only the first few principal components are kept, thus dimensions in each group are reduced. These principal components become the new attributes and are grouped and transformed again in the next step. This step-wise hierarchical approach to dimension reduction, which is the main contribution of this paper, avoids the rank deficiency problem, one would encounter if PCA is applied to all attributes at once.

To highlight the fact that the step-wise procedure described above is a process where transformed attributes become the dimensions at the next step in a hierarchical manner. The steps will be refereed to as levels and the features at each level are referred to as nodes. This provides a visual image of a hierarchical pyramid with higher level having less nodes than the preceding levels as illustrated in Figure 1. Further details of the proposed methodology are provided next.

## 3.1. STAGE 1: GROUPINGS

For illustration of the proposed procedure, consider first a case, where the healthy sample $\mathcal{X}_0$ and the test sample $\mathcal{Y}$ consists of $p = 8$ dimensional vectors. Suppose $\bar{\boldsymbol{y}}$ is the mean vector from sample $\mathcal{Y}$ and $\overline{\boldsymbol{x}}_0$ is the mean from $\mathcal{X}_0$. Without loss of generality, let us assume that the first two attributes are grouped together, then the next two and so on. Thus, the four attribute groups for the eight dimensional case would be $(\boldsymbol{x}_{1,1}, \boldsymbol{x}_{1,2}), (\boldsymbol{x}_{2,3}, \boldsymbol{x}_{2,4}), (\boldsymbol{x}_{2,5}, \boldsymbol{x}_{2,6})$ and $(\boldsymbol{x}_{2,7}, \boldsymbol{x}_{2,8})$. Note that $(\boldsymbol{x}_{t,r}, \boldsymbol{x}_{t,r+1})$ denote the $t^{th}$ group containing attributes $r, r+1$, with $\boldsymbol{x}_{t,r}$ signifying the value associated with attribute $r$.

Since the groupings will be carried out recursively through a hierarchical process, a symbol that denotes the hierarchical level would be needed. Thus, we rewrite the groups as $(\boldsymbol{x}^{(1)}_{t,r}, \boldsymbol{x}^{(1)}_{t,r+1})$, where (1) denotes level one and $t = 1, 2, 3, 4$ and $r = 1, 3, 5, 7$. For any level $i$, the groups then will be denoted by $(\boldsymbol{x}^{(i)}_{t,r}, \boldsymbol{x}^{(i)}_{t,r+1})$ with $t$ and $r$ defined accordingly.

Next $\boldsymbol{X}_t^{(1)} = [\boldsymbol{x}_{t,r}^{(1)} \quad \boldsymbol{x}_{t,r+1}^{(1)}]^T$ be the vector of attribute values in group $t$ and let $\boldsymbol{C}_t^{(1)}$ be the sample correlation matrix estimated from data-samples in $\boldsymbol{\mathcal{X}}_0$. Compute $\boldsymbol{e}_1^{(i)}$, the eigenvector of $\boldsymbol{C}_t^{(\mathbf{1})}$ corresponding to the largest eigenvalue of $\boldsymbol{C}_t^{(\mathbf{1})}$ and let $\boldsymbol{\mathcal{H}}_1^{(t)}$ be the diagonal matrix with the diagonal element representing the estimated standard deviation for every attribute in group $t$.

Since we are computing individual correlation matrices for attributes within each group, and not across all attributes, one can avoid the rank deficiency of the correlation matrix as long as group sizes are controlled. The estimation process is therefore a very effective procedure in high dimensional cases. Moreover, it is possible to remove noisy dimensions in the data systematically. With this information, the second stage of HDR is initiated.

**Remark 3** *As the data is standardized, the covariance matrix used to calculate the principal components is actually the correlation matrix associated with the attributes. See Remark 2 for details*

## 3.2. STAGE 2: TRANSFORMATION

Once, groupings are determined for each group, attributes at any particular level, each group is transformed. The *"Group Transformation"* can be expressed as

$$\boldsymbol{X}_t^{(i+1)} = \boldsymbol{X}_t^{(i)} \boldsymbol{\mathcal{H}}_t^{(i)} \boldsymbol{e}_t^{(i)}, \tag{3}$$

where $\boldsymbol{X}_t^{(i)}$ represents the vector of attribute values for group $t$ at level $i$ corresponding to the reference data $\boldsymbol{\mathcal{X}}_{\mathbf{0}}$. Note that while $\boldsymbol{X}_t^{(i)}$ is a multi-dimensional (two dimensional for this example) vector, $\boldsymbol{X}_t^{(i+1)}$ will be one dimensional. Therefore, once all the groups at this level are transformed, the number of dimensions in the data have been reduced from eight to four.

In a general case, every group need not have only two attributes. In such a case, the data points in a group are given as $\boldsymbol{X}_t^{(i)} = [x_{t,r}^{(i)}, x_{t,r+1}^{(i)}, x_{t,r+2}^{(i)}, \cdots, x_{t,r+\eta(i,t)}^{(i)}]$ where $\eta(i,t)$ is the total number of attributes in group $t$ at level $i$. Note that $i = 1,2,3,\cdots,I$ where $I$ is the total number of levels in the procedure and $t = 1,2,3,\cdots,T^{(i)}$, where $T^{(i)}$ is the total number of groups at level $i$. Also note that $\sum_{t=1}^{T^{(i)}} \eta(i,t) = \eta^{(i)}$, which is the total number of attributes at level $i$. With these notations, the group transformation for a general case is as demonstrated in Eq. (3). However, the dimensions of $\boldsymbol{\mathcal{H}}_t^{(i)}, \boldsymbol{e}_t^{(i)}$, change to $\boldsymbol{\mathcal{H}}_t^{(i)} \in \mathbb{R}^{\eta(i,t)\times\eta(i,t)}$, $\boldsymbol{e}_t^{(i)} \in \mathbb{R}^{\eta(i,t)\times 1}$.

At any particular level $i$, let there be a total of $T^{(i)}$ groups that when transformed result in $T^{(i)}$ nodes at level $i+1$. A closed form expression can be obtained for a one step transformation from level $i \longrightarrow i+1$ as

$$\boldsymbol{X}^{(i+1)} = \boldsymbol{X}^{(i)} diag(\boldsymbol{\mathcal{H}}_1^{(i)}\boldsymbol{e}_1^{(i)}, \cdots, \boldsymbol{\mathcal{H}}_{T^{(i)}}^{(i)}\boldsymbol{e}_{T^{(i)}}^{(i)}). \tag{4}$$

*Derivation: See Appendix.*

Eq. (4) is a generalized expression for the level transformation from $[i \longrightarrow i+1]$. Note that given a correlation matrix $\boldsymbol{C_t^{(i)}}$ with eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{\eta(i,t)} \geq 0$, the eigenvector $\boldsymbol{e}_t^{(i)}$ corresponding to the largest eigenvalue $\lambda_1$ can be obtained as a solution of $\boldsymbol{e}_t^{(i)} = \boldsymbol{C}_t^{(i)^{-1}} \lambda_1 \boldsymbol{C}_t^{(i)}$. The correlation matrix is obtained as, $\boldsymbol{C}_t^{(i)} = N(\boldsymbol{X}_t^{(i)})^T N(\boldsymbol{X}_t^{(i)})$. The standardized normalization, $N(\boldsymbol{X}_t^{(i)})$ is given as

$$N(\boldsymbol{X}_t^{(i)}) = [\boldsymbol{X}_t^{(i)} - \overline{\boldsymbol{X}}_t^{(i)}]\boldsymbol{\mathcal{H}}_t^{(i)}. \tag{5}$$

In Eq. (5), $\overline{\boldsymbol{X}}_{t,r}^{(i)}$ is the mean vector and $\boldsymbol{\mathcal{H}}_t^{(i)} = [diag(\sigma_{t,r+1}^{(i)}, \cdots \sigma_{t,r+\eta(i,t)}^{(i)})]^{-1}$ with $\sigma_{t,r}^{(i)}$ being the standard deviation corresponding to the $r^{th}$ node in the $t^{th}$ group at the $i^{th}$ level with $t = 1,2,\cdots,T^{(i)}$. If $\boldsymbol{\tau}^{(i)} = diag(\boldsymbol{\mathcal{H}}_1^{(i)}\boldsymbol{e}_1^{(i)}, \cdots, \boldsymbol{\mathcal{H}}_{T^{(i)}}^{(i)}\boldsymbol{e}_{T^{(i)}}^{(i)})$, in Eq. (4), the transformation for any observation $p$ from level $[i \longrightarrow i+1]$ (called a *"Level Transformation"*) can be expressed as

$$\boldsymbol{X}^{(i+1)} = \boldsymbol{X}^{(i)}\boldsymbol{\tau}^{(i)}. \tag{6}$$

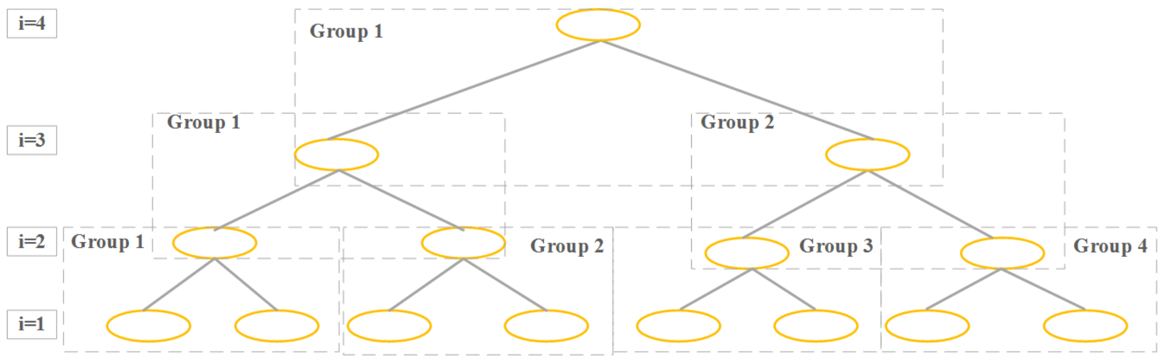


Figure 1. Illustration of the tree

The equation in (6) suggests a transformation from $[i \rightarrow i+1]$ for nodes obtained from the reference sample. Note that in the proposed methodology, groups wise reduction is performed considering the dependencies across groups to be null. However, by performing another step of reduction, these cross-dependencies that were ignored in the previous step can be captured. The main advantage of doing this is that it allows for a methodical process of estimation.

The above introduced steps of grouping and transformation is repeated for a pre-defined number of iterations. A *level transformation,* as expressed in (6) and (8), can be further reduced to result in a closed-form expression for the HDR transformation such that

$$\bar{\boldsymbol{x}}_{\boldsymbol{0}}^{*} = \phi(\boldsymbol{X}^{(1)}) = \boldsymbol{X}^{(1)}\boldsymbol{K}, \tag{7}$$

*Derivation: See Appendix.* The overall HDR procedure is illustrated in Figure 1 for an example.

**Remark 4** *The proposed dimension reducing transformation $\phi$ is completely determined by $\boldsymbol{K}$. Moreover, $\boldsymbol{K}$ is computed from the sample data in $\boldsymbol{X}_{\boldsymbol{0}}$ and is a function of $\hat{\boldsymbol{R}}$ because matrices such as $\boldsymbol{C}_t^{(i)}$ can be shown to be functions of $\hat{\boldsymbol{R}}$. In other words, $\boldsymbol{K}$ is a random matrix that depends on the sample $\boldsymbol{X}_{\boldsymbol{0}}$ through $\hat{\boldsymbol{R}}$ [13]. On the other hand, the results given in Eq.* (7) *are derived assuming $\boldsymbol{K}$ is non-stochastic matrix. Note that $\hat{\boldsymbol{R}}$ is a consistent* *estimator of the true population correlation matrix $\boldsymbol{R}$ and for sufficiently large $\mathcal{M}_0$, $\hat{\boldsymbol{R}} \simeq \boldsymbol{R}$ by law of large numbers. Therefore, $\boldsymbol{K}$ asymptotically achieves a non-stochastic limit. Thus, the result given in Eq.* (7) *can be taken as an asymptotic result, which well approximates the distributions of $(\bar{\boldsymbol{x}}_{\boldsymbol{0}} - \bar{\boldsymbol{y}})\boldsymbol{K}$ for large samples.*

If the test sample at the $i^{th}$ level and the $t^{th}$ group is denoted as

$$Y_t^{(i)} = [y_{t,r}^{(i)}, y_{t,r+1}^{(i)}, \cdots, y_{t,r+\eta(i,t)-1}^{(i)}]$$

, the transformation for the test data can be written as

$$\boldsymbol{Y}^{(i+1)} = \boldsymbol{Y}^{(i)}\boldsymbol{\tau}^{(i)}. \tag{8}$$

The transformation parameters, that is the correlation matrix and the related quantities, are obtained from the reference sample $\boldsymbol{\mathcal{X}}_0$. These parameters are used to transform both the reference sample and the test sample. Similarly, when the data from the test sample is transformed, the final HDR transformation can be written as

$$\bar{\boldsymbol{y}}^* = \phi(\boldsymbol{Y}^{(1)}) = \boldsymbol{Y}^{(1)}\boldsymbol{K} \tag{9}$$

The next step in the procedure is to obtain the distance value. This is done by applying an $L_\rho : \rho > 0$ norm given as

$$\delta(\boldsymbol{\mathcal{X}}_0, \boldsymbol{\mathcal{Y}}) = \|(\boldsymbol{C}_{\boldsymbol{1}}^{\boldsymbol{(I-1)}})^{-1/2}(\bar{\boldsymbol{x}}_{\boldsymbol{0}}^* - \bar{\boldsymbol{y}}^*)\|_\rho^\rho. \tag{10}$$

Eq. (10) is a closed form expression for GDM-HDR, similar to the expression in Eq. (2). One can also note the similarity where the decorrelation step is performed by HDR and the norm is used to induce distance value. From a big data perspective, it is important to analyze the computational requirements in the worst case scenario as shown in Lemma 1.

**Lemma 1** *The worst case computational complexity of calculating GDM-HDR between the mean of the reference sample and the test sample is in the order of*

$$n\Big(\frac{(\eta^{I-1}-1)}{(\eta-1)\eta^{I-1}}O[\eta^3+\eta^2]\Big)+O[\eta^3],$$

*where $\eta$ is the number of attributes in a group.*

**3.2.0.1. Proof:.** See Appendix. □

**Note 1** *Note that in the above given analysis we assume constant group sizes. Furthermore, since $\eta$ represents a constant group size and $\eta << n$, the overall computational complexity would converge to $O[n]$.*

**Remark 5** *There is always a concern of grouping all the important attributes in the same group, leading to premature information loss. To mitigate this, two options can be considered: keep those attributes that are known to be important in separate groups or, if no such information is available, randomize the order of attributes to mitigate the effect.*

The generic methodology presented here can be extended to any size of data with significant computational benefits. Further, a wide variety of applications that generate big data like clustering, medical diagnostics can be addressed in a very effective manner using the proposed methodology. In the following section, a methodology for the application of GDM-HDR to diagnostics and the problem of classification is presented.

## 4. GDM-HDR-BASED FAULT DIAGNOSTICS

To handle complications associated with big data while reducing dimensions and calculating distance values, GDM-HDR was introduced in this paper. In this section, a methodology is outlined to perform real-time detection and isolation of faults using GDM-HDR. This methodology is divided into a total of three stages as shown in Figure 2.

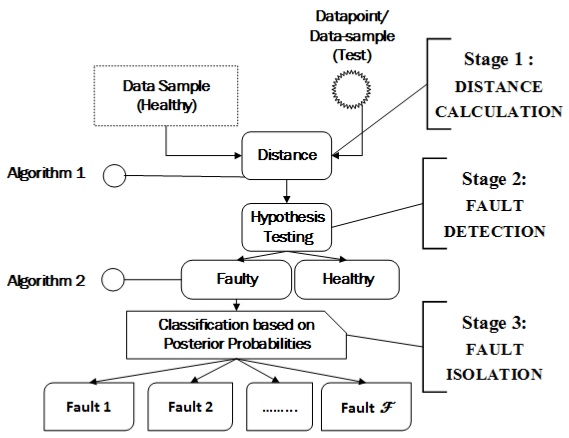


Figure 2. Overview of the proposed methodology

## 4.1. STAGE 1: DISTANCE CALCULATION

First, a sample from the healthy data-set $\boldsymbol{\mathcal{X}}_0 = \{\boldsymbol{X}_p^{(1)} : p = 1, 2, 3, \cdots, \mathcal{M}_0\}$ is chosen. According to the total number of attributes, a tree, as shown in Figure 1, is constructed.

Once a tree is constructed, the distance between any sample $\boldsymbol{\mathcal{Y}}$ and the healthy sample can be calculated by following the hierarchical dimension-reduction procedure as explained in the previous sections. Operational details for this process are provided in Algorithm 1.

**Remark 6** *Note that the fault detection can be performed on a sample if, and only if, Assumption 1 is not violated. Otherwise, it is possible that the proposed technique will not be able to detect the fault. Therefore, in the scenario that Assumption 1 is violated, it is desirable to perform the analysis using only one data point instead of a sample.*

From this point forward, Assumption 1 is relaxed. Analysis is performed on one data point at a time denoted as $\boldsymbol{y}$. In the next section, a methodology is detailed to detect and isolate faults depending on the distance measure.

**Algorithm 1** Generalized Distance Measure; A Hierarchical Dimension Reduction Approach (GDM-HDR)

```
1: Standardize $\boldsymbol{X}^{(1)}$ and $\boldsymbol{Y}^{(1)}$.
2: Randomize the order of the attributes.
3: Generate a tree by grouping attributes (Number of groups can vary at every level)
4: for Level from $i = 1 \rightarrow I$ do
5:     for Groups from $t = 1 \rightarrow T^{(i)}$ do
   Calculate Correlation matrix for $\boldsymbol{X}_t^{(i)}$
6:         Calculate $\boldsymbol{e}_t^{(i)}$
7:         if i is Not Equal to I then
8:             Transform $\boldsymbol{x}_t^{(i+1)} = \boldsymbol{X}_t^{(i)}\boldsymbol{\mathcal{H}}_t^{(i)}\boldsymbol{e}_t^{(i)}, \boldsymbol{y}_t^{(i+1)} = \boldsymbol{Y}_t^{(i)}\boldsymbol{\mathcal{H}}_t^{(i)}\boldsymbol{e}_t^{(i)}$.
9:         else
10:            Calculate Distance value
11:        end if
12:    end for
13: end for
```

Note that the objective in this paper is to first separate all the faulty and healthy data-points based on the distance measure and then classify the faulty points into different faults. In this paper, bootstrap statistical tests are proposed to separate faulty and healthy data-points. Finally a Bootstrap Bayesian classification methodology is proposed to classify these faulty data-points into respective faults.

## 4.2. STAGE 2: FAULT DETECTION

To separate out the faulty data-points from healthy, the idea is to construct two statistical hypotheses. One is the null hypothesis which states that the data is derived from a healthy system. The other is the alternate hypothesis which indicates that the data is derived from a faulty system.

The easiest way to determine a fault is to determine a threshold for $\delta(\boldsymbol{X}_0, \boldsymbol{y})$ for any healthy data point. The $\alpha^{th}$ percentile value distribution obtained by the healthy-condition sample can be considered as such a threshold, where $\alpha$ is chosen by the user. In such scenarios, the distribution of the samples can be accurately determined by sampling with replacement from the data sample. The procedure is known as bootstrap estimation of the distribution and this methodology is formalized next

Let the null hypothesis be given as $H_0 : \delta_y \sim \mathbb{P}_0^*$. The alternate hypothesis is given as $H_a : \delta_y \sim \mathbb{P}_k{}^*, k = 1, 2, 3, \cdots, \mathcal{F}$, where $\mathbb{P}_k{}^*$ is the bootstrap estimate of the underlying distribution. $\delta_y$ is the GDM-HDR value corresponding to the data point $y$ with respect to $\boldsymbol{\mathcal{X}}_0$. If the null hypotheses is rejected, then the data point is obtained from faulty system. On the other hand, if the null hypotheses is not rejected, the system is healthy.

Formally, the first step is to calculate the distance statistic given as shown in Eq. (10). Based on this distance value, $H_0$ at any significance level $\alpha$ is rejected, if

$$\delta_y \geq c_0^*, \tag{11}$$

where $c_0^*$ is the bootstrap critical point or the threshold distance for the healthy scenario. The critical threshold can be computed as

$$P(\delta^* \leq c_0^* | L_0{}^*) = 1 - \alpha, \tag{12}$$

where $\delta^* \in \mathrm{L}_0^*$ and $\mathrm{L}_0^*$ is the bootstrap sample corresponding to all the distance values belonging to the healthy case. To evaluate this critical threshold, one needs to have an approximation of the conditional distribution of the data notationally written as $P(d \leq \delta^* | \mathrm{L}_0^*), \forall d \in \mathbb{R}$.

The distribution is approximated by writing a edge-worth expansions around standard normal given by the following lemma.

**Lemma 2** *Consider a bootstrap sample $L_k{}^*$. Let this be identically and independently sampled from the distance values $\delta(\boldsymbol{x}, \boldsymbol{\mathcal{X}}_k) : \forall \boldsymbol{x} \in \boldsymbol{\mathcal{X}}_k)$, $k = 0, 1, \cdots \mathcal{F}$. The cumulative distribution function (cdf) for the underlying distance distribution for any $\delta^* \in \mathrm{L}_k^*$ can be* *written as*

$$\mathbb{P}_k^* = P(d \leq \delta^* | \mathrm{L}_k^*) = \Phi(\delta^*) - \phi(\delta^*)(1 + \frac{\gamma H_2(\delta^*)}{6\sqrt{B}} + \frac{(\Gamma - 3)H_3(\delta^*)}{24B} + \frac{\gamma^2 H_5(\delta^*)}{72B^2}). \tag{13}$$

*The corresponding probability density function (pdf) is given as*

$$f_k(\delta^*) = \phi(\delta^*)(1 + \frac{\gamma H_2(\delta^*)}{6\sqrt{B}} + \frac{(\Gamma - 3)H_3(\delta^*)}{24B} + \frac{(\gamma^2)H_5(\delta^*)}{72B^2}), \tag{14}$$

*where* $H_1 = \delta^*$, $H_2 = \delta^{*2} - 1$, $H_3 = \delta^{*3} - 3\delta^*$ *and* $H_5 = \delta^{*5} - 10\delta^{*3} + 15\delta^*$ . *In the above equation,* $H_1, H_2, H_3$ *and* $H_5$ *are the Hermite polynomials. Here,* $\phi(x)$ *is the density for the standard normal distribution. And,* $\Phi(x)$ *being the cumulative distribution for standard normal.*

**4.2.0.1. Proof:.** See Appendix. □ The above approximation is applicable to any sampling distribution but is not universal. It imposes certain restrictions on the underlying probability distribution. These restrictions are given by the following assumptions.

**Assumption 2** $E(\|\delta_x\|^4) < \infty, \forall \boldsymbol{x} \in \boldsymbol{\mathcal{X}}_k$

**Assumption 3** *The distribution of the distance values have an absolute continuous component.*

**Assumption 4** *The transformation is continuous, differentiable and has achieved a non stochastic limit.*

The above stated assumptions are acceptable because the amount of data available is large. All these assumptions are satisfied by a normally distributed data. It can be seen here that the proposed approximation considered takes into account fourth order moments. This is very beneficial for capturing any uncertainties in the data. In other words, any deviation from normality can be taken into account, if that is desired.

**Remark 7** *In the case where the distribution is completely symmetric, the first order expansion would be accurate enough for any prediction required.*

The bootstrap sample gives an estimate about the underlying distribution of the distance values. The corresponding approximation of the distribution is given in Lemma 2. Another important aspect is to determine when this approximation converges with the true distribution of the underlying sample as shown in Theorem 1.

**Theorem 1** *Let $L_k{}^*$ be a bootstrap sample and the cdf approximation be given by Lemma 5. The distribution of the sum statistics asymptotically converges with the underlying distribution of the distance values.*

**4.2.0.2. Proof:.** See Appendix □

**Remark 8** *Note that for practical purposes, bootstrap estimation of parameters must be obtained from the sample belonging to each of the faults.*

Once the distribution belonging to the distance sample is known, one can efficiently isolate the faults based on the approximated distribution. This methodology is outlined next.

**Algorithm 2** Classification of faults

```
1:  Assign prior probabilities for each of the class
2:  for any data point $\boldsymbol{y}$ do
3:      if then$\delta(\boldsymbol{X_0}, \boldsymbol{y}) \leq c_0^*$,
4:          The system is healthy
5:      else
6:          for $T = 1 \rightarrow \mathcal{F}$ do
7:              Calculate $Q(T|\boldsymbol{y})$
8:              if $Q(T|\boldsymbol{y})$ is maximum then
9:                  $\mathfrak{r} = T$
10:                 Assign $\boldsymbol{y}$ to class $\mathcal{F}_{\mathfrak{r}}$
11:             end if
12:         end for
13:     end if
14: end for
```

## 4.3. STAGE 3: BOOTSTRAP-BASED FAULT ISOLATION

The Isolation of a fault is performed based on a Bootstrap Bayesian classification methodology detailed below.

Let $\pi_k, k = 1, 2, 3, \cdots \text{W}$ be the prior probabilities for each of the fault classes such that $\sum_{k=0}^{\text{W}} \pi_k = 1$. Let $\boldsymbol{X}_k$ be the data sample for each of the fault classes with the mean vector of the fault classes given as $\overline{\boldsymbol{x}}_k$ for all $k = 1, 2, 3, \cdots, \text{W}$. Also let the probability distribution functions for each fault class be defined by the bootstrap sample of distances of each class and be notationally given by $f_k(\delta_{y,k})$ such that $f_k(\delta_{y,k})$ is defined as shown in Lemma 5, where $\delta(y, \overline{\boldsymbol{x}}_k)$ is notationally written as $\delta_{y,k}$. The idea is to maximize the probability that $y$ belongs to class $T$given by the law of total probability as

$$Q(T|\boldsymbol{y}) = \frac{\pi_T f_T(\delta_{y,T})}{\sum_{k=1}^{\mathcal{F}} \pi_k f_k(\delta_{y,k})}, \tag{15}$$

where $T = 1, 2, 3, \cdots, \mathcal{F}$ . And, $\delta_{\boldsymbol{y},T}$ is the distance for the data-point $\boldsymbol{y}$ with respect to $\mathcal{X}_T$ . Evaluating the maximum probability results in the prediction of a fault.

**Remark 9** *It is important to note that the prior probabilities for the fault classes can be decided based on prior expertise on the data generated by the system under consideration.*

Evaluating $Q(T|\boldsymbol{y})$ for every sample results in a prediction but an approximation of the probability distribution for each of the faults is required and is given by Lemma 2. A step-by-step procedure for fault detection and isolation is detailed in Algorithm 2. In the next section, the result of the proposed methodology on diagnostics and classification are detailed.

# 5. RESULTS AND DISCUSSIONS

All the data-sets used in this study are tabulated first

Table 2. Statistics of the different data-sets used in this paper [31].

| Data-set | Dimensions | Datapoints | Classes |
|---|---|---|---|
| Rolling | 11 | 35000 | 4 |
| Sensorless | 48 | 78000 | 11 |
| Madelon | 500 | 1000 | 2 |
| MNIST | 784 | 72000 | 10 |
| NotMnist | 784 | 81000 | 10 |
| CIFAR-10 | 2072 | 50000 | 10 |
| Gisette | 5000 | 200 | 2 |
| Arcene | 10000 | 100 | 2 |
| Dexter | 20000 | 300 | 2 |

Table 3. Hyper-parameters for different methodologies in this paper.

| Method | Hyper-parameters |
|---|---|
| KNearest Neighbors (KNN) | 3 neighbors |
| Support Vector Machine (SVM) | C = 0.025 |
| Kernel SVM | RBF kernel, gamma = 2, C=1 |
| Desicion Trees (DT) | max depth =5 |
| Random Forest (RF) | max depth=5 estimators=10, max features=1 |
| Shallow Neural Network (SNN) | lr = 0.01, 1 hidden layer, 500 neurons |
| AdaBoost | number of estimators=50, learning rate=1.0 |
| Naive Bayes | No Priors |
| Quadratic Discriminant Analysis (QDA) | No Priors |
| Proposed Bootstrapped Bayesian Classification(BBC) | Equal Priors, bootstrap sample size = 10000 |

## 5.1. DATA-SET

A total of 10 data-sets are used for analysis in this study. The statistics for these data-sets are tabulated in Table 2. A collection of data-sets covering the spectrum of large $n$, small $\mathcal{M}$ and small $n$ large $\mathcal{M}$ are used in this study for evaluation. From Table 2, it can be seen that the first seven data-sets belong to the category of large $\mathcal{M}$ and small $n$ with the last three namely Gisette, Arcene and Dexter data composed of more number of dimension relative to the number of data-points.

All results are averaged over 100 iterations. Furthermore, the corresponding simulations are performed on a computer with 76GB of RAM and a xeon processor.

## 5.2. RESULTS AND DISCUSSIONS- FAULT DIAGNOSTICS

In this section, fault diagnostics performance of the methodology is proposed methodology is demonstrated on the Rolling element bearing dataset and the sensorless diagnostics dataset. In the experiments run below $\rho = 2$ is used for rolling element data-set and $\rho = 0.5$ is used for sensorless drive diagnostics data-sets.

There are a total of 11 attributes in rolling element bearing data-set. Three types of faults are analyzed. Ten thousand data points are randomly sampled from the dataset to constitute the healthy sample and for each of the three fault conditions.

**Remark 10** *The choice of the number 10000 is completely arbitrary here. Also, larger sample size will allow the estimation of transformation parameters to be closer to their true values.*

**Remark 11** *Note that the total number of dimension in rolling element bearing and sensorless drive data-sets are reduced to 5 and 24 respectively.*

**Assumption 5** *At any sampling instant, a complex system generates only one data point and the objective is to determine whether the complex system is at fault or not at that sampling instant. The assumption follows as a consequence of Remark 6.*

The power of the statistical test and the corresponding Type-I error rate is tabulated in Table 4. As expected, the power of the test is high and the corresponding error rates are seen to be low.

To compare the performance of a traditional distance measure like MD and GDM-HDR, statistical tests proposed in this paper are used to detect the faulty data points using the MD values using PCA as the dimension reduction scheme. It is observed in this case that the detection results for GDM-HDR are very close to those obtained by using MD-PCA, which can be verified by comparing column two and three in Table 5.

Table 4. Statistical efficiency of the methodology for the two datasets

| Dataset | significance Level | Threshold | Type I Error | Power |
|---|---|---|---|---|
| Rolling Element | 0.01 | 3.54 | 0.009 | 0.92 |
| | 0.05 | 3.01 | 0.045 | 0.97 |
| | 0.1 | 2.69 | 0.099 | 1.0 |
| Sensorless | 0.01 | 5.42 | 0.010 | 0.91 |
| | 0.05 | 4.84 | 0.049 | 0.93 |
| | 0.1 | 4.55 | 0.097 | 0.94 |

Table 5. Detection results for GDM-HDR and MD+PCA for rolling element bearing data-set and sensorless drive dataset with Bootstrap Bayesian Classification

| Fault | % Correctly Classified | |
|---|---|---|
| | MD+PCA | GDM-HDR |
| Rolling Element | 99.77 | 99.80 |
| Sensorless Drive Diagnostics | 86.34 | 94.35 |

**Remark 12** *Data used for this test are independent of the samples collected previously indicating that important information suggesting a faulty system is not lost while using HDR.*

Sensorless drive data-set is significantly larger than the previously considered data-set with 78000 data points. First, it must be observed that MD fails to measure distance values for this data. It is seen that while calculating MD, one encounters a singular correlation matrix. However, when PCA is used for evaluating distances, one can calculate distance values. On the other hand, when the proposed distance measure along with HDR is used for distance evaluation, such an error is not encountered. The reason is the presence of redundant dimensions in the data. It is seen that GDM-HDR is able to detect faulty data-points as well as MD+PCA in rolling element bearing data-set. On the other hand, the proposed approach outperforms MD+PCA in Sensorless datasets.

Average detection results for the bootstrap test is given in Table 4, whereas the fault isolation results are tabulated in Table 5. One can note from Table 4 that the detection results are acceptable with large values of power and low values of type I error rates. An important aspect of the proposed methodology is to note that GDM-HDR resulted in sub-par results with the rolling element data-set and is optimal with sensorless drive data-set as seen from Table 5. This indicates the ability of GDM-HDR to perform much better in large dimensional cases.

In the next section, relative performance of the proposed dimension reduction methodology is studied on several data-sets including analytical and computational performance for the problem of classification.

## 5.3. RESULTS AND DISCUSSIONS- CLASSIFICATION

All results in this section are averaged over 1000 iterations. Furthermore, the corresponding simulations are performed on a computer with 8GB of RAM and an i7 processor. In this section, the performance of the proposed dimension reduction approach HDR and Bootstrap Bayesian classification is studied without the distance measure.

**5.3.1. Performance Analysis.** All results in this section are summarized using accuracy and Type I error. Accuracy measures the prediction power whereas Type I error error measures the susceptibility of the methodology to noise. Note that high accuracy indicates better results and lower Type-I error indicates better performance. In Table 3, the choice of hyper-parameters corresponding to all the classification methodologies used here are given. Most of the choice of parameters is very standard. First the results of this methodology on all the data-sets is tabulated in Table 6. In this table, the resulting accuracy of BBC with HDR is compared with the case with no HDR.

Note, from the table that the difference in accuracy with and without HDR increases steadily as the dimensions in the corresponding to the data-points increase.

Table 6. Accuracies for the different data-sets with the proposed classification method with or without HDR

| **Data-set** | **Accuracy (HDR)** | **Accuracy (w/o HDR)** |
|---|---|---|
| Rolling | 0.97 | 0.97 |
| Sensorless | 0.90 | 0.90 |
| MNIST | 0.93 | 0.89 |
| NotMnist | 0.93 | 0.80 |
| CIFAR-10 | 0.45 | 0.40 |
| Madelon | 0.91 | 0.79 |
| Arcene | 0.83 | 0.73 |
| Gisette | 0.87 | 0.71 |
| Dexter | 0.82 | 0.69 |

Table 7. Mean accuracies for the various data-sets with different methods using HDR as the dimension reduction methodology with mean false positive rate in parenthesis. The best case result for each row are indicated in bold

| **Data-sets ↓** | **KNN** | **SVM** | **LDA** | **DT** | **RF** | **SNN** | **AdaBoost** | **Naive-Bayes** | **QDA** | **BBC** |
|---|---|---|---|---|---|---|---|---|---|---|
| **Rolling** | **0.99(0.00)** | 0.90(0.02) | 0.97(0.013) | 0.91(0.03) | 0.925(0.012) | 0.70(0.00) | 0.84(0.03) | 0.86 (0.01) | 0.79(0.00) | 0.97(0.01) |
| **Sensorless** | 0.89(0.00) | 0.91(0.01) | 0.84 (0.01) | 0.70(0.01) | 0.65 (0.02) | 0.93(0.00) | 0.25 (0) | 0.73 (0.05) | 0.86(0.06) | **0.93(0.00)** |
| **MNIST** | 0. 94(0.03) | 0.93 (0) | 0.86 (0.02) | 0.67(0.08) | 0.64 (0.02) | **0.96 (0)** | 0.73(0.03) | 0.78 (0.20) | 0.61(0.07) | 0.94(0.08) |
| **NotMnist** | 0.90 (0.01) | 0.87(0.01) | 0.83 (0.03) | 0.72(0.02) | 0.71 (0.03) | 0.90(0.01) | 0.79(0.02) | 0.76 (0.05) | 0.86(0.01) | **0.94(0.01)** |
| **CIFAR-10** | 0.33 (0.08) | 0.38(0.04) | 0.37 (0.05) | 0.26(0.04) | 0.29 (0.09) | 0.50(0.03) | 0.31(0.07) | 0.29 (0.07) | 0.36(0.02) | **0.64(0.03)** |
| **Madelon** | 0.49 (0.28) | 0.57(0.21) | 0.57 (0.21) | 0.79(0.08) | 0.55 (0.18) | 0.59(0.20) | 0.59(0.19) | 0.59 (0.23) | 0.51(0.24) | **0.55(0.08)** |
| **Arcene** | 0.70 (0) | 0.83 (0) | 0.79 (0) | 0.60 (0) | 0.69 (0) | 0.79 (0) | 0.63 (0) | 0.45 (0) | 0.61 (0) | **0.86 (0)** |
| **Dexter** | 0.49 (0) | 0.57 (0) | 0.57 (0) | 0.79 (0) | 0.52 (0) | 0.53 (0) | 0.59 (0) | 0.59 (0) | 0.51 (0) | **0.99(0.01)** |
| **Gisette** | 0.94 (0.03) | 0.98(0.01) | 0.90 (0.05) | 0.91(0.04) | 0.73 (0.07) | 0.97(0.01) | 0.96(0.01) | 0.73 (0.25) | 0.63(0.12) | **0.99(0.01)** |

Table 8. Average classification accuracy (Avg. Class. Acc.) rates for various dimensions reduction techniques (Dim-Red) and classifiers using Sensorless Drive Diagnostics data set.[32]

| **Dim-Red ↓** | KNN | SVM | DT | RF | LDA | SNN | AdaBoost | Naive-Bayes | QDA |
|---|---|---|---|---|---|---|---|---|---|
| PCA | 0.52 | 0.52 | 0.52 | 0.45 | 0.45 | 0.52 | 0.18 | 0.45 | 0.53 |
| ISOMAP | 0.67 | 0.46 | 0.43 | 0.32 | 0.39 | 0.58 | 0.18 | 0.42 | 0.49 |
| LLE | 0.25 | 0.08 | 0.15 | 0.18 | 0.18 | 0.08 | 0.18 | 0.19 | 0.19 |
| FastICA | 0.54 | 0.53 | 0.52 | 0.48 | 0.44 | 0.58 | 0.28 | 0.45 | 0.53 |
| FA | 0.592 | **0.91** | 0.66 | 0.48 | 0.41 | 0.94 | 0.18 | 0.75 | 0.86 |
| **HDR** | 0.90 | **0.90** | **0.94** | **0.65** | **0.59** | **0.95** | **0.18** | **0.76** | **0.87** |

.

The main observation to make here is the fact that the proposed methodology is optimal in the case where the total number of data-points are significantly less than the total dimensions in the data as seen by the results from Arcene, Gisette and dexter data-set. Next, the proposed classification methodology along with HDR is compared for accuracies with the most traditional classification methods and the results are tabulated in Table 7. It can be again observed that the proposed methodology performs optimally for data-sets with large dimensions, again bolstering the observation that was made before.

It can be noted that, for the CIFAR-10 and Madelon data-set, the proposed methodology does not performs very well. The lack of performance on CIFAR-10 data-set may be attributed to the complicated nature of the data-set. Furthermore, random forests and decision tree fail with complicated data-sets.

It has been observed from the results that the proposed methodology performs very well in across data-sets and across classification approaches. In the next section relative performance of the proposed dimension reduction on big data relative to other traditional dimension reduction approaches is studied.

## 5.4. SCALABILITY ANALYSIS OF HDR

First, synthetic data is generated with increasing number of attributes and the computational overhead of different dimension reduction approaches are compared. It is seen that, with the proposed methodology that is HDR, the computational overhead is least in Figure 3. In fact we note that in places where methods such as PCA, LLE and ISOMAP does not provide reasonable level of efficiency the proposed methodology is able to perform efficiently as seen from the results in Table 8 and Figure 3.

Note from Table 8, that the proposed dimension reduction approach achieved highest performance when compared with traditional dimension reduction approaches. Moreover, note that tests are performed on a 48 dimensional data.

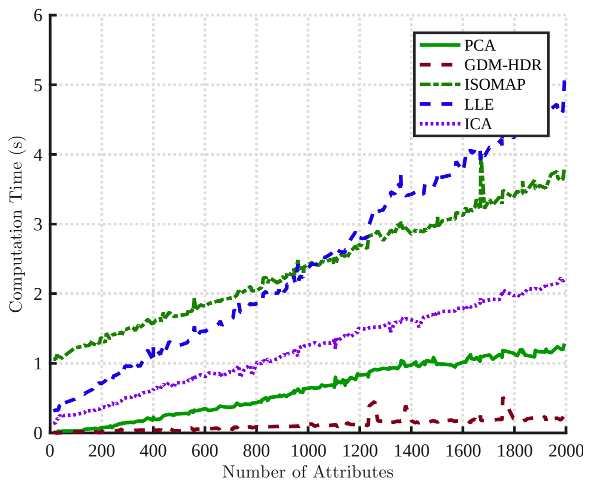


Figure 3. Trends of computational time with respect to the attribute number for different dimension reduction methodologies

This is because, most of the methods such as PCA, LLE, ISOMAP fail due to computational overload in data-sets such as arcene, dexter and gisette. However, the proposed approach can allow for reasonable detection for these data-sets which is highly satisfactory in case of big data situations. As a consequence, a fair comparison could not be shown on data-sets such as Arcene, dexter and gisette and a 48 dimensional dataset was chosen.

A reasonable question that arises here is whether the proposed dimension reduction approach suffers from a relatively large information loss when compared with standard approaches such as PCA. In other words, we want to understand would each step in HDR contributes to information loss in a cumulative manner? A mitigating factor in the proposed approach is that after the first step, principal components from groups are combined, thereby bringing in information about their correlations. These correlations can be treated as a proxy for the correlations between variables across the adjoining groups, which were ignored in the previous step.

Table 9. Averaged $R^2$values obtained by regressing the components from HDR against the components derived from PCA by sample size based on 1000 simulation runs.

| **# of sample points $\rightarrow$** | **4096** | **2048** | **1024** | **512** | **256** | **128** | **56** | **10** | **5** |
|---|---|---|---|---|---|---|---|---|---|
| Average $R^2$ value | 0.8607 | 0.8623 | 0.8646 | 0.8697 | 0.8778 | 0.8907 | 0.9172 | 0.9957 | 1 |

Table 10. Averaged $R^2$ values obtained by regressing the components from HDR against the components derived from PCA by sample size and the number of PCA components.

| **# of Components$^{\downarrow}$** | **# of sample points** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **4096** | **2048** | **1024** | **512** | **256** | **128** | **56** | **10** | **5** |
| 5 | 0.8914 | 0.8884 | 0.9011 | 0.9067 | 0.9147 | 0.8811 | 0.9447 | 1 | 1 |
| 10 | 0.8499 | 0.8503 | 0.8588 | 0.8578 | 0.8570 | 0.8710 | 0.9096 | 1 | 1 |
| 15 | 0.8404 | 0.8412 | 0.8439 | 0.8651 | 0.8637 | 0.8813 | 0.9302 | 1 | 1 |
| 20 | 0.8806 | 0.8870 | 0.8849 | 0.8927 | 0.8899 | 0.9114 | 0.9330 | 1 | 1 |

In order to understand how much information is lost by the HDR relative to a single step PCA, a regression approach that regresses the first $k$ principal components of PCA against $k$ components selected by HDR is performed. The average $R^2$ value for $k$ components from HDR and PCA over 1000 simulation run is tabulated in Table 9. Note that in Table 9, the value of $k$ is chosen at random. Furthermore, it must be observed that if HDR retains most of the information then the $R^2$ value obtained from the regressors should be close to one.

Next, in Table 10, these averaged $R^2$ values are broken down by the number of components such that $k = 5, 10, 15, 20$. Results from Table 9 and 10 indicate that almost 90% of the information contained in the $k$ set of components are explained by components derived from HDR.

## 6. CONCLUSION

In this paper, generalized distance measure with a novel hierarchical dimension reduction approach (GDM-HDR) was proposed. The proposed methodology was applied to the problem of diagnostics.

The performance of the proposed distance measure is acceptable in cases where a standard distance measure such as MD does not perform well. It is seen from the results that GDM-HDR is more optimal in large dimensional scenarios compared to the small scale cases. It was shown that the proposed measure, when paired with classification, can effectively perform fault detection and isolation for big data.

It is observed, both from the approach and the case study, that expert domain knowledge is required to make informative decisions in the preprocessing stage. A certain amount of data is required for establishing reference and samples for fault classes. The proposed classification methodology takes into account uncertainties in the data which allows for better estimation of data with a non-normal distribution.

It has been shown both empirically and theoretically that the worst case complexity for the proposed distance evaluation is orders of magnitude less than that of MD.

Admirable performance is observed when the HDR is compared with popular dimension reduction methodologies. In fact, it is seen that HDR is able to outperform standard dimension reduction approaches in the high dimensional cases. Looking ahead, we aim to extend the proposed methodology for nonlinear dependencies.

## ACKNOWLEDGMENT

This research was supported in part by NSF I/UCRC award IIP 1134721 and Intelligent Systems Center.

## APPENDIX

**6.0.0.1. Proof:.** [Derivation of the level transformation] To derive this transformation, first, express the largest principle component corresponding to the first group at level $i$ consisting of $\eta(i,1)$ nodes (which is also the first node at level $(i+1)$) as

$$x_1^{(i+1)} = \boldsymbol{X_1^{(i)}}\boldsymbol{\mathcal{H}_1^{(i)}}\boldsymbol{e}_1^{(i)}. \tag{16}$$

The largest principal component corresponding to the last group at level $i$ consisting of $\eta(i,T^{(i)})$ nodes can be expressed as

$$x_{T^{(i)}}^{(i+1)} = \boldsymbol{X_{T^{(i)}}^{(i)}}\boldsymbol{\mathcal{H}_{T^{(i)}}^{(i)}}.\boldsymbol{e}_{T^{(i)}}^{(i)} \tag{17}$$

The overall transformation from the level $i$ to level $i+1$ can be written in vector form as

$$= \\ [\boldsymbol{X_1^{(i)}}\boldsymbol{\mathcal{H}_1^{(i)}}\boldsymbol{e}_1^{(i)} \quad \boldsymbol{X_2^{(i)}}\boldsymbol{\mathcal{H}_2^{(i)}}\boldsymbol{e}_2^{(i)} \quad \cdots \quad \boldsymbol{X_{T^{(i)}}^{(i)}}\boldsymbol{\mathcal{H}_{T^{(i)}}^{(i)}}\boldsymbol{e}_{T^{(i)}}^{(i)}], \tag{18}$$

Relabel $[x_1^{(i+1)} \quad x_2^{(i+1)} \quad \cdots \quad x_{T^{(i)}}^{(i+1)}]$ as $\boldsymbol{X}^{(i+1)}$ and $[\boldsymbol{X_1^{(i)}} \quad \boldsymbol{X_{p,2}^{(i)}} \quad \cdots \quad \boldsymbol{X_{p,T^{(i)}}^{(i)}}]$ as $\boldsymbol{X}_p^{(i)}$, where $\boldsymbol{X}_p^{(i)}$ and $\boldsymbol{X}_p^{(i+1)}$ represent all nodes at level $i$ and $i+1$, respectively. As a result of some rearranging, it follows that

$$\boldsymbol{X}_p^{(i+1)} = \boldsymbol{X}_p^{(i)}[diag(\boldsymbol{\mathcal{H}_1^{(i)}}\boldsymbol{e}_1^{(i)}, \boldsymbol{\mathcal{H}_2^{(i)}}\boldsymbol{e}_2^{(i)}, \cdots, \boldsymbol{\mathcal{H}_{T^{(i)}}^{(i)}}\boldsymbol{e}_{T^{(i)}}^{(i)})]. \tag{19}$$

which is the level transformation as shown in Eq. (4) □

**6.0.0.2. Proof:.** [Derivation of the closed form solution] To achieve this, let the transformation from level one to two be expressed as,

$$\begin{aligned} \boldsymbol{X}^{(2)} &= \boldsymbol{X}^{(1)}\boldsymbol{\tau}^{(1)}, \\ \boldsymbol{Y}^{(2)} &= \boldsymbol{Y}^{(1)}\boldsymbol{\tau}^{(1)}. \end{aligned} \tag{20}$$

Similarly the transformations from the second to the third level are given as

$$\begin{aligned} \boldsymbol{X}^{(3)} &= \boldsymbol{X}^{(2)}\boldsymbol{\tau}^{(2)}, \\ \boldsymbol{Y}^{(3)} &= \boldsymbol{Y}^{(2)}\boldsymbol{\tau}^{(2)}. \end{aligned} \tag{21}$$

From (20), (21), one can write

$$\begin{aligned} \boldsymbol{X}^{(3)} &= \boldsymbol{X}^{(1)}\boldsymbol{\tau}^{(1)}\boldsymbol{\tau}^{(2)}, \\ \boldsymbol{Y}^{(3)} &= \boldsymbol{Y}^{(1)}\boldsymbol{\tau}^{(1)}\boldsymbol{\tau}^{(2)}. \end{aligned} \tag{22}$$

Generalizing this for $I$ levels, the transformed data at the penultimate level can be written as

$$\begin{aligned} \boldsymbol{X}^{(I-1)} &= \boldsymbol{X}^{(1)}\boldsymbol{\tau}^{(1)}\boldsymbol{\tau}^{(2)}\cdots\boldsymbol{\tau}^{(I-2)} \\ \boldsymbol{Y}^{(I-1)} &= \boldsymbol{Y}^{(1)}\boldsymbol{\tau}^{(1)}\boldsymbol{\tau}^{(2)}\cdots\boldsymbol{\tau}^{(I-2)}. \end{aligned} \tag{23}$$

where the transformation matrix $\boldsymbol{\tau}^{(\boldsymbol{i})} : i = 1, 2, 3, \cdots (I-2)$ is given as a block diagonal matrix. Let the tree based transformation be designated as $\phi$, one can conclude that the transformation from level 1 to level $I-1$ is given as

$$\phi(\boldsymbol{X}^{(1)}) = \boldsymbol{X}^{(1)}\boldsymbol{K}, \tag{24}$$

such that $\boldsymbol{K} = \boldsymbol{\tau}^{(1)}\boldsymbol{\tau}^{(2)}\cdots\boldsymbol{\tau}^{(I-2)}$. The result in Eq. (10) directly follows from this. □

**6.0.0.3. Proof:.** [Computational Complexity Lemma]

At the first step in GDM-HDR, the total number of attributes in the tree are paired into groups of $\eta$. As a result there are a total of $n/\eta$ groups at the bottom most level. Let there be $I$ levels in the tree. The total number of groups until the penultimate level of the tree assuming equal group sizes can be counted as

$$n\sum_{i=1}^{I-1}\frac{1}{\eta^i} = n\Big(\frac{(\eta^{I-1}-1)}{(\eta-1)\eta^{I-1}}\Big). \tag{25}$$

Eigen-decomposition is performed in each of these groups. The most computationally expensive operation is that of calculating the covariance matrix and eigenvalue decomposition. This process has a complexity of $O[p^3+p^2]$, where $p$ is the number of dimension involved. As a result the worst case upper bound for dimension reduction in one group can be counted as $O[\eta^3+\eta^2]$. We can therefore write the total approximate asymptotic complexity from the lowest level until the penultimate level for one data-point as

$$n\frac{(\eta^{I-1}-1)}{(\eta-1)\eta^{I-1}}O[\eta^3+\eta^2] \tag{26}$$

At the penultimate level, the most expensive operation during norm calculation is that of the inversion of the covariance matrix which has the complexity of $O[p^3]$. Since, $p=\eta$, the complexity can be written as $O[\eta^3]$. If we add the operations in Eq. (26) and (25) then the total asymptotic computational complexity of GDM-HDR can be achieved and this is given as,

$$n\Big(\frac{(\eta^{I-1}-1)}{(\eta-1)\eta^{I-1}}O[\eta^3+\eta^2]\Big)+O[\eta^3]. \tag{27}$$

This completes our proof $\square$

**6.0.0.4. Proof:.** [Approximate distribution of GDM-HDR] Define a sum statistic from $\mathbb{L}_k^*$ such that $S_B = \sigma^{-1}\sqrt{B}(\delta_x - \overline{\delta_x}) : \forall \delta_x \in \mathbb{L}_k^*$, where $\overline{\delta}_x$ is the empirical mean and $\sigma_B$ is the sample standard deviation for the bootstrap sample. Next, write the empirical characteristic function as

$$\Psi_B(t) = E\{\exp^{itS_B}\}, \forall t \in \mathbb{R}. \tag{28}$$

Write the power series expansion of the exponential function, simplify and rearrange to achieve

$$\Psi_B(t) = e^{-t^2/2}\{1 + B^{-1/2}p_1(it) + B^{-1}p_2(it) + B^{-2}p_3(it) + \cdots\} \tag{29}$$

where $p_1(it) = (it)^3\gamma/6, p_2(it) = (it)^4(\Gamma{-}3)/24, p_3(it) = (it)^6(\gamma^2)/72$ with $\gamma = E\{\delta(\boldsymbol{x}, \mathcal{X}_k)\}^3$ and $\Gamma = E\{\delta(\boldsymbol{x}, \mathcal{X}_k)\}^4$ being the third and fourth order moments.

Under the assumption that the fourth order moment (Assumption 2) is finite, the density of the underlying distribution can be obtained using inversion formula of characteristic function [33]. It follows that

$$\begin{aligned} g(\delta_x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-it\delta_x}(e^{-t^2/2}\{1 + B^{-1/2}p_1(it)^3 + B^{-1}p_2(it) + B^{-2}p_3(it) \\ + \cdots\})dt \end{aligned} \tag{30}$$

Since $S_B$ is asymptotically normal by CLT, the characteristic function would converge to normality [33], it follows that

$$f_k(\delta_x) = g(\delta_x) = \phi(\delta_x)\left(1 + \frac{\gamma H_2}{6\sqrt{B}} + \frac{(\Gamma - 3)H_3}{24B} + \frac{\gamma^2 H_5}{72B^2}\right) \tag{31}$$

Integrating the above equation and simplifying on the similar lines as [34] reveals

$$G(\delta_x) = \Phi(\delta_x) - \phi(\delta_x)\left(1 + \frac{\gamma H_2}{6\sqrt{B}} + \frac{(\Gamma - 3)H_3}{24B} + \frac{\gamma^2 H_5}{72B^2}\right) \tag{32}$$

In the two equations mentioned above, $H_1 = E[\delta_x]$, $H_2 = E[\delta_x]^2 - 1$, $H_3 = E[\delta_x]^3 - 3E[\delta_x]$ and $H_5 = E[\delta_x]^5 - 10E[\delta_x]^3 + 15E[\delta_x]$ are the Hermite polynomial. And $\phi(\delta_x)$ is the density for the standard normal distribution with $\Phi(\delta_x)$ being the conditional density for the standard normal distribution. □

**6.0.0.5. Proof:.** [Bootstrap Convergence Proof] Let the distance statistic be represented as $\delta(\boldsymbol{X}_k, \boldsymbol{x})$ and the corresponding sample and its bootstrap counterpart be given as $L^*$ and $L$ respectively. Let the size of the sample be $B$ and the bootstrap distribution be represented as $\mathbb{P}^*$. By Lemma 2, the approximations of $\mathbb{P}$ can be written as

$$\mathbb{P}(\delta_y) = \Phi(\delta_y) - \phi(\delta_y)\left(1 + \frac{\gamma H_2}{6\sqrt{B}} + \frac{(\Gamma - 3)H_3}{24B} + \frac{\gamma^2 H_5}{72B^2}\right). \tag{33}$$

The Edgeworth expansion of the bootstrap counterpart can be written as

$$\mathbb{P}^*(\delta_y) = \Phi(\delta_y^*) - \phi(\delta_y^*)\left(1 + \frac{\gamma H_{2*}}{6\sqrt{B}} + \frac{(\Gamma - 3)H_{3*}}{24B} + \frac{\gamma^2 H_{5*}}{72B^2}\right), \tag{34}$$

where $\delta_y \in L$, $\delta_y^* \in L^*$. $H_1 = E[\delta_x]$, $H_2 = E[\delta_x]^2 - 1$, $H_3 = E[\delta_x]^3 - 3E[\delta_x]$ and $H_5 = E[\delta_x]^5 - 10E[\delta_x]^3 + 15E[\delta_x]$ in the above two equations such that $E[.]$ represents the expected value. By subtracting Eq. (33) from (34) and rearranging

$$\mathbb{P}(\delta_y) - \mathbb{P}^*(\delta_y) = \phi(\delta_y)\left[\left(\frac{\gamma(H_2 - H_{2*})}{6\sqrt{B}} + \frac{(\Gamma - 3)(H_3 - H_{3*})}{24B} + \frac{\gamma^2(H_5 - H_{5*})}{72B^2}\right)\right] \tag{35}$$

Asymptotically and under the finiteness of four moments, it is easy to see that $\mathbb{P}(\delta_y) - \mathbb{P}^*(\delta_y)$ tends to zero at the rate of $O[\frac{1}{n^{\frac{1}{2}}}]$. which proves that, the bootstrap distribution of the sample asymptotically converges with the true distribution in probability. $\square$

## REFERENCES

[1] J. Fan, F. Han, H. Liu, Challenges of big data analysis, National science review 1 (2) (2014) 293–314.

[2] R. Clarke, H. W. Ressom, A. Wang, J. Xuan, M. C. Liu, E. A. Gehan, Y. Wang, The properties of high-dimensional data spaces: implications for exploring gene and protein expression data, Nature Reviews Cancer 8 (1) (2008) 37–49.

[3] C. Giraud, Introduction to high-dimensional statistics, Vol. 138, CRC Press, 2014.

[4] X. Wu, X. Zhu, G.-Q. Wu, W. Ding, Data mining with big data, ieee transactions on knowledge and data engineering 26 (1) (2014) 97–107.

[5] C. P. Chen, C.-Y. Zhang, Data-intensive applications, challenges, techniques and technologies: A survey on big data, Information Sciences 275 (2014) 314–347.

[6] L. Zhang, J. Lin, R. Karim, Sliding window-based fault detection from high-dimensional data streams, IEEE Transactions on Systems, Man, and Cybernetics: Systems.

[7] C.-K. Ing, T. L. Lai, M. Shen, K. W. Tsang, S.-H. Yu, Multiple testing in regression models with applications to fault diagnosis in big data era, Technometrics (just-accepted).

[8] A. Soylemezoglu, S. Jagannathan, C. Saygin, Mahalanobis-taguchi system as a multi-sensor based decision making prognostics tool for centrifugal pump failures, IEEE Transactions on Reliability 60 (4) (2011) 864–878.

[9] D. Ververidis, C. Kotropoulos, Information loss of the mahalanobis distance in high dimensions: Application to feature selection, IEEE transactions on pattern analysis and machine intelligence 31 (12) (2009) 2275–2281.

[10] C. C. Aggarwal, A. Hinneburg, D. A. Keim, On the surprising behavior of distance metrics in high dimensional space, in: International Conference on Database Theory, Springer, 2001, pp. 420–434.

[11] D. Francois, V. Wertz, M. Verleysen, The concentration of fractional distances, IEEE Transactions on Knowledge and Data Engineering 19 (7) (2007) 873–886.

[12] F. Khan, D. Kari, I. A. Karatepe, S. S. Kozat, Universal nonlinear regression on high dimensional data using adaptive hierarchical trees, IEEE Transactions on Big Data 2 (2) (2016) 175–188.

[13] R. A. Johnson, D. W. Wichern, Applied multivariate statistical analysis, Vol. 5, Prentice hall Upper Saddle River, NJ, 2002.

[14] C. Xu, Y. Zhang, R. Li, X. Wu, On the feasibility of distributed kernel regression for big data, IEEE Transactions on Knowledge and Data Engineering 28 (11) (2016) 3041–3052.

[15] K. P. Adragni, E. Al-Najjar, S. Martin, S. K. Popuri, A. M. Raim, Group-wise sufficient dimension reduction with principal fitted components, Computational Statistics 31 (3) (2016) 923–941.

[16] Z. Guo, L. Li, W. Lu, B. Li, Groupwise dimension reduction via envelope method, Journal of the American Statistical Association 110 (512) (2015) 1515–1527.

[17] A. D. Ward, G. Hamarneh, The groupwise medial axis transform for fuzzy skeletonization and pruning, IEEE Transactions on pattern analysis and machine intelligence 32 (6) (2010) 1084–1096.

[18] J. Zhou, J. Wu, L. Zhu, Overlapped groupwise dimension reduction, Science China Mathematics 59 (12) (2016) 2543–2560.

[19] I. Jolliffe, Principal component analysis, Wiley Online Library, 2002.

[20] R. A. Johnson, D. W. Wichern, Applied multivariate statistical analysis, Vol. 4, Prentice hall Englewood Cliffs, NJ, 1992.

[21] I. K. Fodor, A survey of dimension reduction techniques (2002).

[22] M. Balasubramanian, E. L. Schwartz, The isomap algorithm and topological stability, Science 295 (5552) (2002) 7–7.

[23] S. David, S. Ruey, et al., Independent component analysis via distance covariance, Journal of the American Statistical Association.

[24] S. T. Roweis, L. K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.

[25] D. L. Donoho, C. Grimes, Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data, Proceedings of the National Academy of Sciences 100 (10) (2003) 5591–5596.

[26] M. Belkin, P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, Neural computation 15 (6) (2003) 1373–1396.

[27] H. Feng, et al., Gene classification using parameter-free semi-supervised manifold learning, IEEE/ACM transactions on computational biology and bioinformatics 9 (3) (2012) 818–827.

[28] J. Shawe-Taylor, N. Cristianini, Kernel methods for pattern analysis, Cambridge university press, 2004.

[29] J. Gui, S.-L. Wang, Y.-K. Lei, Multi-step dimensionality reduction and semi-supervised graph-based tumor classification using gene expression data, Artificial intelligence in medicine 50 (3) (2010) 181–191.

[30] P. C. Mahalanobis, On the generalized distance in statistics, Proceedings of the National Institute of Sciences (Calcutta) 2 (1936) 49–55.

[31] M. Lichman, UCI machine learning repository (2013).

[32] R. Silipo, I. Adae, A. Hart, M. Berthold, Seven techniques for dimensionality reduction, KNIME.

[33] P. Hall, Inverting an edgeworth expansion, The Annals of Statistics 11 (2) (1983) 569–576.

[34] P. Hall, The bootstrap and Edgeworth expansion, Springer Science & Business Media, 2013.

# II. A MULTI-STEP NONLINEAR DIMENSION-REDUCTION APPROACH WITH APPLICATIONS TO BIG DATA

R. Krishnan

Department of Electrical & Computer Engineering

Missouri University of Science and Technology

Rolla, Missouri 65409–0050

Email: krm9c@mst.edu

V.A. Samaranayake

Department of Mathematics & Statistics

Missouri University of Science and Technology

Rolla, Missouri 65409–0050

Email: vsam@mst.edu

S. Jagannathan

Department of Electrical & Computer Engineering

Missouri University of Science and Technology

Rolla, Missouri 65409–0050

Email: sarangap@mst.edu

## ABSTRACT

In this paper, a novel dimension-reduction approach is presented to overcome challenges such as nonlinear relationships, heterogeneity, and noisy dimensions typical with big data. In the proposed approach, dimensions in the data are reduced while considering the data in batches. For each batch, the *p* attributes are organized into groups at random. Next, to systematically remove redundant and noisy dimensions from the data, each group is independently mapped into a low dimensional space via a parametric mapping. The group-wise mapping parameters are then estimated using the power method, where the distance covariance is approximated with a low rank singular value decomposition (SVD). The transformed attributes are reorganized into groups based on the magnitude of their respective singular values. The singular value based group-wise organization and subsequent reduction process is performed until a user-defined criterion on singular values is satisfied. The process of dimension reduction is repeated for all batches. To address the problem of heterogeneity, a novel approach is utilized to aggregate the batch-wise transformation parameter estimates. Overall performance is demonstrated with extensive simulation analysis on classification by employing nine big data-sets. The proposed approach is shown to outperform other standard dimension-reduction methods with a 24 % improvement.

## 1. INTRODUCTION

Big data-sets have become increasingly common in fields such as statistics, machine learning, security, and fault diagnostics [1]. These data-sets are characterized by large $p$, the number of attributes, and/or large $n$, the number of datapoints. Unlike in small $n$, small $p$ scenarios, where it is easy to analyze the whole data-set at once, in most practical big-data situations, all the data is not available at once. Furthermore, when these new samples become available, they may differ from the available samples, which represents the problem of statistical heterogeneity. Big data-sets are also accompanied by incomplete and redundant dimensions where complications such as noise and spurious correlations commonly exists [2, 3]. More importantly, several data-sets such as CiFAR-10 [4], MNIST[5], and others, display nonlinear relationships among dimensions.

The accumulation of these challenges deteriorates performance during analysis which necessitating the development of new techniques [6]. One way to mitigate the redundancy in dimensions and spurious correlations is to extract relevant attributes from the data.

In light of this, a multi-step nonlinear dimension-reduction approach (NDR) is proposed in this paper. Here, a dependence measure known as distance covariance [7] is used to incorporate nonlinear relationships among attributes while reducing dimensions to filter noisy dimensions from big-data. The use of distance covariance [8] enables the efficient capture of nonlinear dependencies. However, when $p >> n$, matrices quantifying pairwise dependencies through distance covariance can be rank-deficient due to concentration of norm in high-dimensional spaces [3]. To handle this, a group-wise dimension-reduction procedure is proposed where the distance covariance matrix is computed for each group separately.

In NDR, the $p$ attributes from a sample of data are first organized into random groups at the initial stage. Next, each group is considered independently and dependencies among the attributes are measured using distance covariance [7]. Then, attributes in each group are transformed into a low-dimensional space via a parametric mapping using the singular value decomposition (SVD) of distance covariance. Based on a user-defined parameter determining the amount of variation, these transformation parameters are tuned to approximate a low rank SVD. Once, each group of attributes is transformed, the new attributes are reorganized into groups and transformed again. However, to ensure that the transformed attributes with maximum variation are kept in separate groups, the group-wise organization is performed using the magnitude of the singular values of distance covariance instead of at-random.

Since the number of dimensions required to capture a specific proportion of information is unknown beforehand, an adhoc process is required to specify the number of dimensions to be extracted. To address the problem, a singular value-based user criterion is introduced here to determine the number of dimensions to be extracted. Therefore, the group-wise organization and transformation process described above is repeated until the criterion, controlling the degree of information loss is satisfied. To update the transformation parameters when new data is observed, a matrix factorization-based, batch-wise updates are proposed to refine the estimates of distance covariance and the corresponding SVD.

The general performance of NDR is demonstrated on ten data-sets, some of which fall into the category of big data, using standard classification methods. One of the data-sets involves two million datapoints with twenty thousand dimensions. A logistic regression scheme using Hamilton Monte Carlo (HMC) optimization for tuning the regression parameters is one of the methods utilized for diagnostics and/or classification.

The contributions of this paper therefore include (1) development of a multi-step, group-wise, dimension-reduction approach using distance covariance to handle nonlinear relationships and noisy dimensions; (2) design of a matrix-factorization based mechanism to update distance covariance and the corresponding SVD in situations where data is available in batches; (3) design of a generic group-wise organization procedure and explicit stopping criterion to control information loss; (4) demonstration of the efficacy of NDR using ten data-sets covering the problem of classification and fault diagnostics.

The rest of the paper is organized as follows. Section 2 describes preliminaries and establishes motivations for the paper. Section 3 describes the proposed dimension-reduction methodology with the algorithms and methodology for fault diagnostics and/or classification. Section 4 outlines simulation results for the paper while Section 5 provides conclusions for the paper.

## 2. PRELIMINARIES AND RELATED WORKS

In this section, the notations for the paper are discussed then the problem under consideration is formulated. For general notations, consider $\mathbb{R}$ to represent the set of real numbers and let the set of natural numbers be written as $\mathbb{N}$. Throughout the paper, matrices and vectors are denoted by boldface with lowercase symbols used for vectors and uppercase symbols for denoting matrices. These notations are also summarized in Table 1. Squared Euclidean norm for vectors and squared Frobenius norm for matrices are denoted by $||.||$.

Table 1. Notations

| **Notation** | **Meaning** |
|---|---|
| $\boldsymbol{X}$ | Data-set |
| $\mathcal{F}$ | Number of faults/classes in the system. |
| $(n, p)$ | Number of sample points in the training set, the dimensionality of data |
| $k$ | Index for faults/classes |
| $m, l$ | Indexes throughout the paper |
| g(.) | Characteristic function |
| $\boldsymbol{A}, \boldsymbol{B}, \tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}}$ | Distance matrices and their double-centered counterpart |
| $v_n^{a,b}$ | Sample distance covariance |
| $r_n^{a,b}$ | Sample distance correlation |
| $\boldsymbol{C}$ | Distance correlation matrix |
| $\lambda$ | Eigenvalues |
| $\boldsymbol{J}$ | Batch update of distance correlation matrix |
| $\boldsymbol{K}$ | Updated transformation matrix for batch updates |
| $M$ | Number of batches in the data |
| $(i) : i = 1, 2, \cdots I$ | Step index in the dimension-reduction process represented by the superscript |
| $j$ | Index for attributes at each step |
| $t : \quad 1, 2, 3, \cdots T^{(i)}$ | Group index at any particular step $(i)$ while $T^{(i)}$ is the total number of groups at step $(i)$ |
| $\kappa_t^{(i)}, \eta_t^{(i)}$ | Estimated cardinality of the low dimensional space for group $t$ and step $i$ and $\eta_t^{(i)}$ is the number of attributes in each group |
| $\boldsymbol{P}^{(i)}$ | Transformation matrix at every step |
| $\boldsymbol{Q}^{(i)}$ | Shuffling matrix at every step |
| $\alpha$ | Amount of variation to be captured in a group |

## 2.1. PROBLEM FORMULATION

Consider a scenario where data can arise under one of the $\mathcal{F}$ categories. Suppose $\boldsymbol{y}$ is a datapoints, with $p$ attributes, obtained under one of these categories. In classification problems, we want to determine the probability of a particular datapoints $\boldsymbol{y}$ belonging to category $k, k = 1, 2, \cdots, \mathcal{F}$ given a knowledge base (training data-set) $\boldsymbol{\mathcal{X}}$. Thus, we are interested in

$$p(\boldsymbol{y} \in \boldsymbol{\Psi}_k | \boldsymbol{\mathcal{X}}), \tag{1}$$

where $\boldsymbol{\Psi}_k$ with $k = 1, 2, 3, \cdots, \mathcal{F}$ denotes the population representing category $k$. Note that a generalized regression problem that can be used to evaluate these probabilities can be defined as

$$p(\boldsymbol{y} \in \boldsymbol{\Psi}_k | \boldsymbol{\mathcal{X}}) = f(\boldsymbol{\theta}; \boldsymbol{y}), \tag{2}$$

where the training data-set be denoted by $\boldsymbol{\mathcal{X}}$ with $\{\boldsymbol{\mathcal{X}}\} = \cup_{k=1}^{\mathcal{F}} \boldsymbol{\mathcal{X}}_k$, where $\boldsymbol{\mathcal{X}}_k$ is the sample points belonging to category $k$. Each datapoints in $\boldsymbol{\mathcal{X}}$ is collected across $p$ attributes. Furthermore, the parameters of the regression are denoted by $\boldsymbol{\theta}$.

**Remark 13** *Since the classification problem described in Eq.* (2) *is generic, all methodologies that use some form of a parametric map can be considered for analysis. Examples of such methods include neural networks and logistic regressions [9].*

While classifying datapoints in big data scenarios using the problem formulation in Eq. (2), the number of dimensions in the data adversely affects the classification efficiency. To observe this effect, consider classification with the MNIST data-set. A three-layer neural network is used for classification with the MNIST digits-recognition data-set [10] where redundant dimensions are introduced synthetically into the data. When the number of redundant dimensions in the data increases, the accuracy falls drastically, as observed from Figure 1.

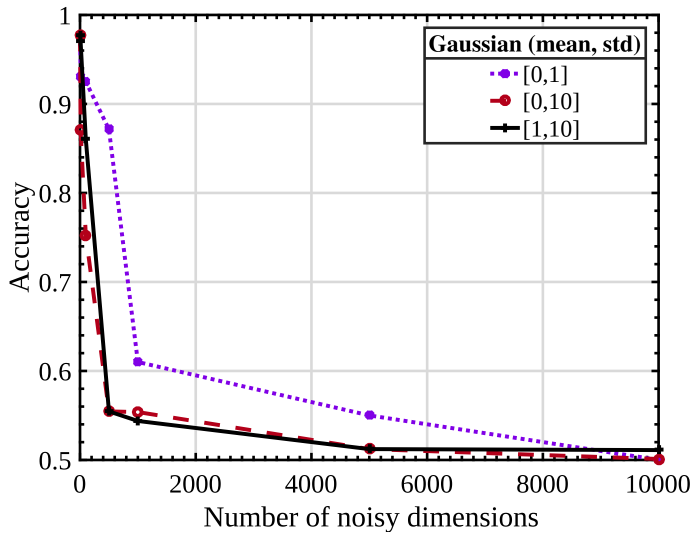


Figure 1. Accuracy with respect to an increase in the number of noisy dimensions in the data.

To mitigate this effect, it is desirable to reduce the number of dimensions prior to classification [6]. However, due to the challenges posed by the high dimensional space, the process of dimension reduction is nontrivial. In traditional dimension-reduction approaches such as principal component analysis (PCA) [11] and factor analysis [12] are frequently used for feature extraction. In these methodologies [11], [12], the data is linearly projected into a lower dimensional space using the eigen-decomposition of correlation matrices. However, efficient projection is only possible if the dependence relationships are estimated accurately.

Two problems arise while evaluating dependence relationships in big data. First, imperfect estimation is observed in the presence of nonlinear relationships among dimensions, especially when Pearson correlation/covariance matrices are used [11, 12, 13]. Second, a full-rank Pearson correlation/covariance matrix cannot be obtained when large $p$, small $n$ data-sets are involved [3]. As a consequence, efficient feature extraction cannot be ensured when using methodologies such as those described in [11, 12].

Therefore, a measure of dependency characterizing independence in a more general framework is required. In a seminal paper by Professor Szekely [7], one such dependence measure known as distance covariance (DC) was proposed. For simplicity, consider the training data to have $n$ observations with only two attributes ($p = 2$). Thus, each vector in $\boldsymbol{X}$ can be written as $[\boldsymbol{a} \quad \boldsymbol{b}]$. Let $\boldsymbol{a} = [a_1 \quad \cdots \quad a_n]^T$ and $\boldsymbol{b} = [b_1 \quad \cdots \quad b_n]^T$. The population version of distance covariance can be written as follows.

**Definition 1** *DC between random vectors $\boldsymbol{a} \in \mathbb{R}^{n\times1}$ and $\boldsymbol{b} \in \mathbb{R}^{n\times1}$ under finiteness of first moment is given as*

$$\nu^{\boldsymbol{a},\boldsymbol{b}} = \|g_{\boldsymbol{a},\boldsymbol{b}} - g_{\boldsymbol{a}}g_{\boldsymbol{b}}\|, \tag{3}$$

*where $g_{\boldsymbol{a},\boldsymbol{b}}$ is the joint characteristic function and $g_{\boldsymbol{a}}$ and $g_{\boldsymbol{b}}$ are the respective marginals.*

It can be clearly seen that $\nu^{\boldsymbol{a},\boldsymbol{b}}$ is zero iff $g_{\boldsymbol{a},\boldsymbol{b}} = g_{\boldsymbol{a}}g_{\boldsymbol{b}}$, which is only true when both $\boldsymbol{a}$ and $\boldsymbol{b}$ are independent. In other words, $\boldsymbol{\nu}^{\boldsymbol{a},\boldsymbol{b}}$ is a non-negative and sure measure of independence for any kind of relationship among two random vectors. As a consequence, DC is appropriate for characterizing nonlinear relationships with big data. A sample estimate of DC is described as follows.

Let $\boldsymbol{A}$ and $\boldsymbol{B}$ represent a matrix of pairwise distances between every pair of elements in vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ respectively. Note that $\boldsymbol{A} = \{\boldsymbol{A}_{m,l} : m, l = 1, 2, \cdots n\}$ and $\boldsymbol{B} = \{\boldsymbol{B}_{m,l} : m, l = 1, 2, \cdots n\}$ with $\boldsymbol{A}_{m,l} = (a_m - a_l)^2$ and $\boldsymbol{B}_{m,l} = (b_m - b_l)^2$. Let $(.)_{m.}$ denote the $m^{th}$ row in (.) and $(.)_{.l}$ denotes the $l^{th}$ column in (.). Next, double center $\boldsymbol{A}$ and $\boldsymbol{B}$ such that $\tilde{\boldsymbol{A}}$ and $\tilde{\boldsymbol{B}}$ represent the double centered matrices where

$$\begin{aligned}
\tilde{\boldsymbol{A}}_{m,l} &= \begin{cases} \boldsymbol{A}_{m,l} - \frac{1}{n}\boldsymbol{1}_n^T(\boldsymbol{A})_{.l} - \\ \frac{1}{n}(\boldsymbol{A})_{m.}\boldsymbol{1}_n + \frac{1}{n^2}\boldsymbol{1}_n^T\boldsymbol{A}\boldsymbol{1}_n & m \neq l \\ 0 & m == l \end{cases} \\
\tilde{\boldsymbol{B}}_{m,l} &= \begin{cases} \boldsymbol{B}_{m,l} - \frac{1}{n}\boldsymbol{1}_n^T(\boldsymbol{B})_{.l} - \\ \frac{1}{n}(\boldsymbol{B})_{m.}\boldsymbol{1}_n + \frac{1}{n^2}\boldsymbol{1}_n^T\boldsymbol{B}\boldsymbol{1}_n & m \neq l \\ 0 & m == l, \end{cases}
\end{aligned} \tag{4}$$

where $\boldsymbol{1}_n$ is a column vector of ones of length $n$.

**Definition 2** *Let the second-order moment of* $\boldsymbol{a}$ *and* $\boldsymbol{b}$ *be finite. The squared sample DC is defined as*

$$\nu_n^{\boldsymbol{a},\boldsymbol{b}} = \frac{1}{n^2}\mathbf{1}_n^T(\tilde{\boldsymbol{A}} \odot \tilde{\boldsymbol{B}})\mathbf{1}_n, \tag{5}$$

*where* $\odot$ *represents the Hadamard product and* $\mathbf{1}_n$ *represents a vector of ones with size n. The squared sample distance correlation can be written as*

$$r_n^{\boldsymbol{a},\boldsymbol{b}} = \frac{\nu_n^{\boldsymbol{a},\boldsymbol{b}}}{\sqrt{\nu_n^{\boldsymbol{a},\boldsymbol{a}}\nu_n^{\boldsymbol{b},\boldsymbol{b}}}}. \tag{6}$$

Note that the distance covariance as defined in [7] allows for estimating independence between multivariate samples. The objective in this paper is to systematically remove redundant dimensions; therefore, we look at pairwise dependencies among attributes.

The objective of this paper is to present a dimension reduction methodology that addresses the challenges such as distance concentration and spurious correlation.

## 2.2. RELATED WORKS

Several dimension-reduction techniques such as Isomap [14], LLE [15], Hessian LLE [16], Laplacian eigen-maps [17] and its variants [18] including kernel PCA [19], have been introduced to handle nonlinear relationships. These methodologies discover the intrinsic geometric structure of high-dimensional data. In [3], the authors show that high dimensional spaces are sparse and suffer from distance concentration [3]. This challenge makes the discovery of intrinsic geometric structure nontrivial. In NDR, distance covariance is utilized to evaluate nonlinear relationships while the sparsity of the high-dimensional space is addressed by evaluating dependencies in groups.

More recently, distance covariance has been utilized as a tool to perform effective independent component analysis in [20] where dimensions are decorrelated in the presence of nonlinear relationships and has been shown to be effective. However, there is a need to estimate the complete distance covariance matrix and its inverse. Therefore, this method would still suffer from improper estimation [6, 3].

To address the problem of computational disadvantages and imperfect estimation in large dimensional scenarios, divide-and-conquer mechanisms for dimension reduction were proposed in [21, 22, 23, 24], where the authors have shown that in many cases, known application-specific relationships and natural groupings can be exploited for efficient dimension reduction. However, such approaches [21, 22, 23, 24], require prior knowledge of these relationships to enable organization, which may not be available in big data scenarios.

Similar to [21, 22, 23, 24], a divide-and-conquer approach is presented in this paper, too. However, in contrast to [21, 22, 23, 24], NDR allows the use of SVD results to perform group-wise organization. In addition, if information about such relationships is available, the generic organization structure proposed here allows for the incorporation of this information at the first step.

In traditional dimension reduction approaches [11, 12, 13, 14, 20, 15, 16, 17, 18, 19, 21, 22, 23, 24], the underlying principle is to perform one-step mapping from a higher dimensional space to a lower dimensional space. First, such an approach is computationally unfriendly when the dimension of the original data is very large. Moreover, it is also susceptible to improper estimation due to the presence of large amounts of noise and redundancies [25]. In contrast, a multi-step dimension-reduction approach is introduced in the proposed NDR. Furthermore, while standard dimension-reduction approaches require the practitioner to specify the number of dimensions to be extracted from the data. In contrast, the user only specifies the percentage of information to be retained in NDR and NDR can estimate the number of dimensions to be extracted from the data.

In the dimension-reduction approaches mentioned earlier [11, 12, 13, 14, 20, 15, 16, 17, 18, 19, 21, 22, 23, 24], there is no mechanism to incorporate new information into the analysis structure when new data is available. In such scenarios [11, 12, 13, 14, 20, 15, 16, 17, 18, 19, 21, 22, 23, 24], one would either be required to retrain the dimension reducing transformation or ignore all the previous samples of data [11, 12, 13], which is inefficient.

Although parameter aggregation methods facilitating the update of Pearson correlation and/or covariance [12] with the corresponding SVD [26] exist, they are focused towards computational efficiency when all the data is available. Since it is not the case in big data scenarios, these methods may be inefficient. Furthermore, these parameter aggregation methods are not applicable with distance covariance due to the difference in construction between Pearson correlation and distance covariance. Thus, in contrast with [11, 12, 13, 14, 15, 16, 17, 19, 21, 22, 23, 24, 26], novel batch-wise update procedures suited to both computational efficiency and parameter aggregation methods for use with distance covariance are derived in this paper.

One potential application of NDR is in the field of visualization, where approaches such as TSNE and deep-autoencoders [27, 28] have shown to be very promising and effective. First of all, deep-autoencoders and TSNE are efficient for visualization, however, they [27, 28] require multiple passes through the data for visualization and therefore computationally scale with the number of datapoints. On the other hand, NDR is computationally efficient and can perform reasonable visualization when large dimensional data is available.

More importantly, popular visualization methods [27, 28] optimize a cost function for effectiveness. Due to this, one may be at a disadvantage if all the data is not available at once and there is no mechanism to incorporate new information. On the other hand, with batch processing updates proposed in NDR, one is able to incorporate new information as new data is available. Furthermore, the changes in the data structure over the course of time can be observed with NDR.

# 3. NONLINEAR DIMENSION-REDUCTION

Encountering large dimensions in big-data applications is a major issue faced by data analysts. To mitigate this challenge, in each step of the dimension-reduction procedure, we propose to split the number of dimensions in the data into groups. Next, we reduce the dimensions in each group via a transformation where nonlinear relationships are considered using distance covariance. Finally, we merge the newly extracted dimensions. The split and merge process is repeated until a predefined condition is satisfied.

Another issue of concern in big-data situations is that data may arrive in batches. Moreover, sometimes it is not practical to analyze the data all at once. To address this challenge, the dimension-reduction process proposed herein can handle data arriving in batches by aggregated information. This refinement of the previous estimates with information from new data also allows us to address the issue of heterogeneity. A flowchart of the overall methodology is shown in Figure 2.

## 3.1. MULTI-STEP GROUP-WISE DIMENSION REDUCTION

For illustration, first consider a simple case, where the sample $\boldsymbol{\mathcal{X}} = [\boldsymbol{x}_1 \quad \cdots \quad \boldsymbol{x}_n]^T$, with $\boldsymbol{x}_1$ representing the first observation in $\boldsymbol{\mathcal{X}}$ with eight dimensions such that p=8, and $\boldsymbol{x}_n$ represents the $n^{th}$ sample point in the data-set. Alternatively, the data in the first step can be denoted by a matrix $\boldsymbol{X}^{(1)} = [\boldsymbol{x}_1^{(1)} \quad \cdots \quad \boldsymbol{x}_8^{(1)}]$ such that $\boldsymbol{x}_j^{(1)}$ represents the $j^{th}$ column in $\boldsymbol{X}^{(1)}$. In other words, $\boldsymbol{x}_j^{(1)}$ represents the vector of values taken by the $j^t$ dimension for all the $n$ observations. The proposed methodology is composed of a sequence of steps with $i = 1, \cdots, I$, where each step can be described in two stages, the grouping stage and the transformation stage.

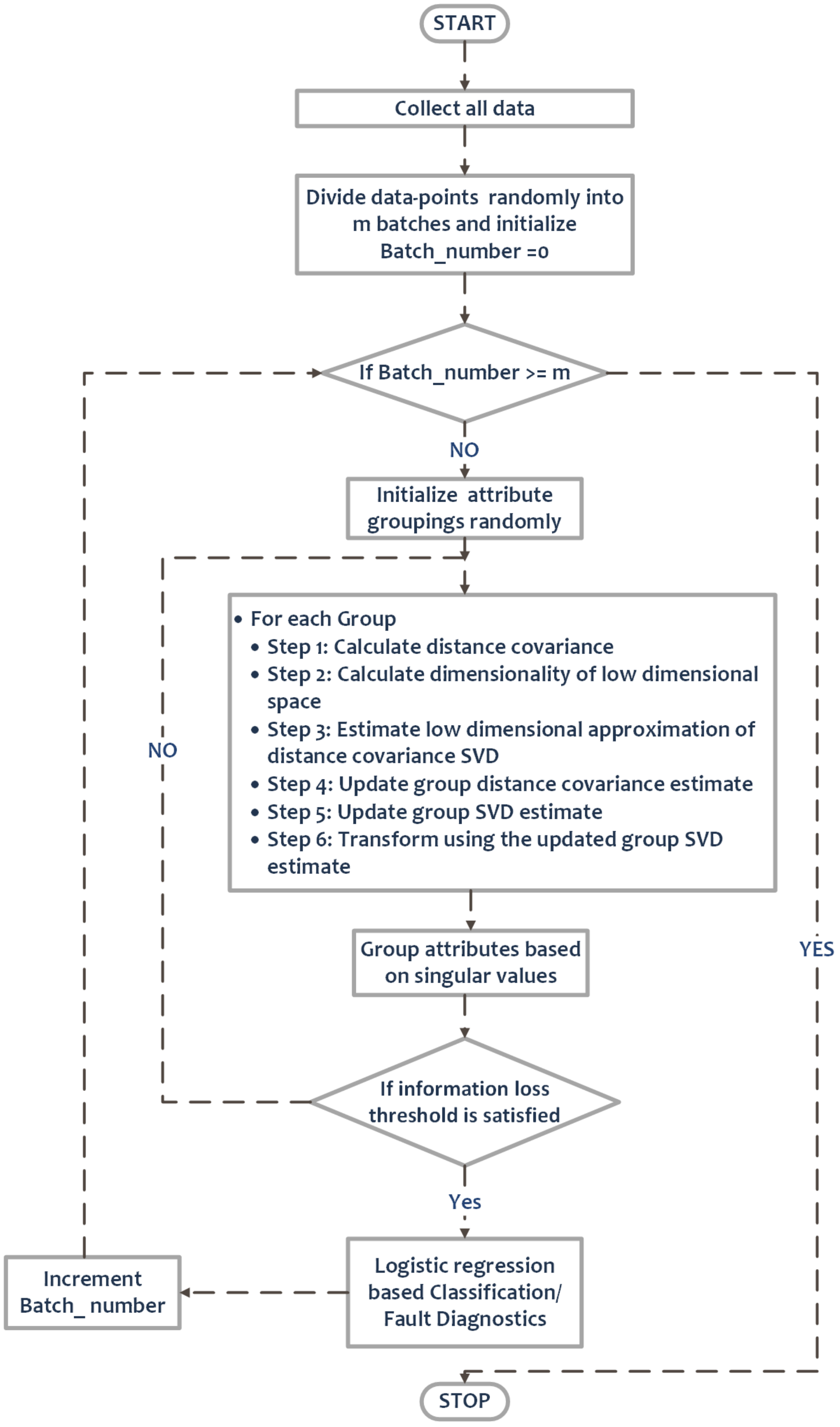

START
Collect all data
Divide data-points randomly into m batches and initialize Batch_number =0
If Batch_number >= m
NO
Initialize attribute groupings randomly
• For each Group
• Step 1: Calculate distance covariance
• Step 2: Calculate dimensionality of low dimensional space
• Step 3: Estimate low dimensional approximation of distance covariance SVD
• Step 4: Update group distance covariance estimate
• Step 5: Update group SVD estimate
• Step 6: Transform using the updated group SVD estimate
NO
Group attributes based on singular values
YES
If information loss threshold is satisfied
Yes
Logistic regression based Classification/ Fault Diagnostics
Increment Batch_ number
STOP

Figure 2. Overview of the dimension-reduction methodology

**3.1.1. Stage 1: Groupings When $i = 1$.** The first stage consists of creating groups from the data directly. The initial groupings are performed randomly. Potentially, if some information about the structure of data is available, then the initial groupings could be constructed using it. As an example, we will consider the eight-dimensional case for analysis.

Therefore, at the start of the dimension-reduction approach, the eight attributes are grouped as $(1,2),(3,4),(5,6),(7,8)$ such that $(\boldsymbol{x}_1^{(1)}, \boldsymbol{x}_2^{(1)})$ form group $t = 1$ and $(\boldsymbol{x}_3^{(1)}, \boldsymbol{x}_4^{(1)})$ form group $t = 2$ and so on.

Let $\boldsymbol{X}_t^{(i)}$ denote the attributes at the $i^{th}$ step and $t^{th}$ group. Note that $i = 1,2,3,\cdots,I$, where $I$ is the total number of steps in the dimension-reduction process, and $t = 1,2,3,\cdots,T^{(i)}$, where $T^{(i)}$ is the total number of groups at step $i$. With these notations and the initial grouping, the transformation stage is initiated.

**3.1.2. Stage 2: Transformation.** For each group, let $\boldsymbol{C}_1^{(1)}$ be the DC matrix estimated for the pair $(\boldsymbol{x}_1^{(1)}, \boldsymbol{x}_2^{(1)})$ using Eq. (5), thus yielding

$$\boldsymbol{C}_1^{(1)} = \begin{bmatrix} \mathcal{V}_n^{\boldsymbol{x}_1^{(1)},\boldsymbol{x}_1^{(1)}} & \mathcal{V}_n^{\boldsymbol{x}_1^{(1)},\boldsymbol{x}_2^{(1)}} \\ \mathcal{V}_n^{\boldsymbol{x}_2^{(1)},\boldsymbol{x}_1^{(1)}} & \mathcal{V}_n^{\boldsymbol{x}_2^{(1)},\boldsymbol{x}_2^{(1)}} \end{bmatrix}.$$

As observed, the matrix $\boldsymbol{C}_1^{(1)}$ is symmetric and of size $2 \times 2$. The diagonal elements represent individual variances, and the off-diagonal elements represent pairwise relationships. By using distance covariance to measure relationships among dimensions, the strength of the relationships are captured even when the two dimensions are nonlinearly related.

The objective is to transform the data into a new dimensional space such that redundancies among attributes are minimized and the variance captured by the reduced dimension is maximized [11]. Therefore, let $\boldsymbol{C}_t^{(i)}$ denote the DC matrix for group $t$ and define the singular value decomposition of $\boldsymbol{C}_t^{(i)}$ as

$$\boldsymbol{C}_t^{(i)} = \boldsymbol{U}_t^{(i)}\boldsymbol{\Sigma}_t^{(i)}\boldsymbol{V}_t^{(i)^T}. \tag{7}$$

Note that $\lambda_{t,1} \geq \lambda_{t,2} \geq \cdots \geq \lambda_{t,\eta_t^{(i)}} \geq 0$ denotes the eigenvalues of $\boldsymbol{C}_t^{(i)}$ and $\boldsymbol{\Sigma}_t^{(i)} = diag(\sqrt{\lambda_{t,1}}, \sqrt{\lambda_{t,2}}, \cdots \sqrt{\lambda_{t,\eta_t^{(i)}}})$ with $\boldsymbol{U}^{(i)}$ and $\boldsymbol{V}^{(i)}$ being the orthonormal transformations, which are the eigenvectors of $\boldsymbol{C}_t^{(i)}$.

To perform the group-wise transformation, the size of the projected space is determined depending on a threshold set by the practitioner. This threshold is denoted as $\alpha/100$, where $\alpha$ refers to the percentage of information that must be retained from each group. To this end, the eigenvalues are first normalized to sum to one. Next, the largest $\kappa_t^{(i)}$ eigenvalues are selected such that their cumulative sum is greater than $\alpha/100$ and the rest of the dimensions are discarded. Using the selected eigenvalues and the corresponding eigenvectors, the data in each group is projected to a new space where these dimensions explain a minimum of $\alpha\%$ variation in the data. In this paper, this projection is performed using a low-rank approximation of SVD. The problem of lower rank estimation can be solved by using techniques such as coordinate ascent or the power method. These iterative mechanisms are readily available in the literature [29]. One such estimation process is given in Algorithm 4.

Formally, let $\kappa_t^{(i)}$ be the number of dimensions that are derived from this group such that $\kappa_t^{(i)}$ dimensions explain $\alpha$ % of the variation in the group data. Thus, $\kappa_t^{(i)}$ is given as

$$\kappa_t^{(i)} = min\{j : \sum_{m=1}^{j} \frac{\lambda_{t,m}}{\sum_{l=0}^{\eta_t^{(i)}} \lambda_{t,l}} \geq \frac{\alpha}{100} \\ , j = 1, \cdots, \eta_t^{(i)}\}, \tag{8}$$

where $\eta_t^{(i)}$ represents the number of attributes in group $t$ at level $(i)$. Depending on $\kappa_t^{(i)}$, a low dimensional approximation corresponding to $\boldsymbol{C}_t^{(1)}$ is computed [29] and denoted by $\hat{\boldsymbol{C}}_t^{(i)} = \hat{\boldsymbol{U}}_t^{(i)}\hat{\boldsymbol{\Sigma}}_t^{(i)}(\hat{\boldsymbol{V}}_t^{(i)})^T$ with $\hat{\boldsymbol{V}}_t^{(i)}$ and $\hat{\boldsymbol{U}}_t^{(i)}$ representing the eigenbasis with rank $\kappa_t^{(i)}$. Further, $\hat{\boldsymbol{\Sigma}}_t^{(i)}$ represents a diagonal matrix with $\kappa_t^{(i)}$ eigenvalues. As a result, the transformation parameters that best extract $\alpha$% variation from the group are obtained through $\hat{\boldsymbol{U}}^{(i)}$.

With the current example, let $\alpha = 95\%$ and assume that in each group, the number of dimensions explaining 95% variation is one. Then, the original data is transformed into four new data-vectors using the eigenvectors corresponding to each of the largest eigenvalues found in each group. Thus, dimensions are reduced from eight to four.

Formally, let $\boldsymbol{P}_t^{(1)} = \hat{\boldsymbol{U}}_t^{(1)}$ denote the transformation parameters obtained from the approximation in each group such that

$$\hat{\boldsymbol{X}}^{(2)} = \{\hat{\boldsymbol{X}}_t^{(2)}, \forall t = 1, 2, 3, 4\}, \tag{9}$$

where $\hat{\boldsymbol{X}}_t^{(2)} = \boldsymbol{X}_t^{(1)}\boldsymbol{P}_t^{(1)}$ and $\boldsymbol{X}_t^{(1)} \in \mathbb{R}^{n\times 2}$, and after transformation $\hat{\boldsymbol{X}}_t \in \mathbb{R}^{n\times 1}$. Next, we aggregate data from all the groups after transformation to achieve

$$\hat{\boldsymbol{X}}^{(2)} = [\hat{\boldsymbol{X}}_1^{(2)} \quad \hat{\boldsymbol{X}}_2^{(2)} \quad \hat{\boldsymbol{X}}_3^{(2)} \quad \hat{\boldsymbol{X}}_4^{(2)}].$$

In general, one can describe the cumulative transformation from step $i$ to $i+1$ as follows. Let $\boldsymbol{P}_t^{(i)}$ be the projection matrix which forms the eigenbasis for the reduced dimensions in group $t$ and step $(i)$. Thus, we can write,

$$\begin{aligned}
\hat{\boldsymbol{X}}^{(i+1)} &= \{\boldsymbol{X}_t^{(i)}\boldsymbol{P}_t^{(i)}, t = 1, 2, \cdots T^{(i)}\}, \\
&= [\boldsymbol{X}_1^{(i)}\boldsymbol{P}_1^{(i)}, \boldsymbol{X}_2^{(i)}\boldsymbol{P}_2^{(i)}, \cdots, \boldsymbol{X}_{T^{(i+1)}}^{(i)}\boldsymbol{P}_{T^{(i+1)}}^{(i)}] \\
&= [\boldsymbol{X}_1^{(i)} \quad \boldsymbol{X}_2^{(i)} \quad \cdots \quad \boldsymbol{X}_{T^{(i+1)}}^{(i)}] \\
&\qquad diag(\boldsymbol{P}_1^{(i)}, \boldsymbol{P}_2^{(i)}, \cdots, \boldsymbol{P}_{T^{(i+1)}}^{(i)}) \\
&= \boldsymbol{X}^{(i)}\boldsymbol{P}^{(i)},
\end{aligned}$$

where $\boldsymbol{X}^{(i)} = [\boldsymbol{X}_1^{(i)} \quad \boldsymbol{X}_2^{(i)} \quad \cdots \quad \boldsymbol{X}_{T^{(i+1)}}^{(i)}]$ with $\boldsymbol{P}^{(i)} = diag(\boldsymbol{P}_1^{(i)}, \boldsymbol{P}_2^{(i)}, \cdots, \boldsymbol{P}_{T^{(i+1)}}^{(i)})$. For each group, the sum of the eigenvalues greater than the threshold for the groups is known. Under the assumption that the information contained in a dimension is proportional to the eigenvalues, an average of the eigenvalues from each group provides an estimate about the amount of information that is retained. The objective is to control the information loss while reducing the dimensions. Thus, if $\kappa_t^{(i)}$ in all groups becomes equal to $\eta_t^{(i)}$ for each group, the process is stopped. It can be interpreted that the dimensions cannot be reduced further without losing more information, which acts as a stopping criterion.

**Remark 14**

1. *Increasing the value of $\alpha$ increases the number of dimensions that move to the next step.*

2. *This process of transformation, according to the magnitude of the eigenvalues, allows the proposed methodology to remove noise and redundant dimensions. This is, however under the assumption that the attributes with smaller eigenvalues are noisy and redundant. This assumption is justified in the unsupervised dimension-reduction case. Note that this is the primary assumption behind popular unsupervised dimension-reduction approaches such as PCA, FA, etc. Relaxing this assumption would require feedback from the detection methodology, which is not the goal of this paper.*

3. *When a redundant dimension is encountered in a group, the proposed selection procedure allows for the removal of dimensions with zero or minimal contribution to the variation.*

4. *During the initial group-wise organization, the dependencies across the groups are ignored. However, by performing another step of group-wise reduction, these cross-dependencies, ignored in the previous step are captured. Repeated application of the proposed procedure can capture these dependencies to some extent. This allows for a methodical process of capturing relationships with algorithmic simplicity.*

Assuming that the stopping criterion was not satisfied in the first step, the group-wise organization must be performed again and Stage 1 of the methodology is reinitiated.

**3.1.3. Stage 1: Groupings When $i > 1$ .** For illustration, let the stopping criterion be unmet after the first step of dimension- reduction. Next, denote the set of transformed datapoints from the first step in the dimension-reduction procedure such that these transformed datapoints are ordered in descending order. These new dimensions are grouped according to the strength of eigenvalues. In other words, the dimensions with smaller eigenvalues are grouped with dimensions with larger eigenvalues. By doing this, we ensure that the dimensions with large variance are not placed in one group and discarded at the dimension-reduction step.

The principle behind this group-wise organization is to form groups according to their contribution to the overall variation. Specifically, in the example used above, it can be seen that attributes one and two contribute more to variance. On the other hand, attributes three and four contribute less. The objective therefore is to keep one and two in separate groups. This structure reduces the premature loss of attributes that contributes more to the variation thereby mitigating information loss. With this line of thought, two new groups can be constructed for our example. The first attribute is assigned to the first group and the second attribute is assigned to the second group. Next, the third attribute is assigned to the first group and the fourth attribute is assigned to the second group.

In a general case, let $\boldsymbol{X}^{(i+1)}$ contain observations with transformed dimensions and ordered according to the eigenvalues obtained in the previous step. Consider $T^{(i+1)}$ to be the total number of groups constructed from $\hat{\boldsymbol{X}}^{(i+1)}$. Similar to the example earlier, the first $T^{(i+1)}$ attributes are assigned to each of the $T^{(i+1)}$ groups. Then, the next $T^{(i+1)}$ attributes are assigned to each of the $T^{(i+1)}$ groups. The process is continued till all the attributes in

$\hat{\boldsymbol{X}}^{(i+1)}$ are assigned to a group. The overall process results in the groups given as

$$\begin{aligned}
\boldsymbol{X}_1^{(i+1)} &= [\hat{\boldsymbol{X}}_1^{(i+1)}, \hat{\boldsymbol{X}}_{1+1T^{(i+1)}}^{(i+1)}, \cdots, \hat{\boldsymbol{X}}_{1+\eta_1^{(i+1)}T^{(i+1)}}^{(i+1)}] \\
\boldsymbol{X}_2^{(i+1)} &= [\hat{\boldsymbol{X}}_2^{(i+1)}, \hat{\boldsymbol{X}}_{2+1T^{(i+1)}}^{(i+1)}, \cdots, \hat{\boldsymbol{X}}_{2+\eta_2^{(i+1)}T^{(i+1)}}^{(i+1)}] \\
&\vdots \\
\boldsymbol{X}_{T^{(i+1)}}^{(i+1)} &= \qquad [\hat{\boldsymbol{X}}_{T^{(i+1)}}^{(i+1)}, \hat{\boldsymbol{X}}_{T^{(i+1)}+1T^{(i+1)}}^{(i+1)}, \cdots, \\
&\qquad\qquad \hat{\boldsymbol{X}}_{T^{(i+1)}+\eta_{T^{(i+1)}}^{(i+1)}T^{(i+1)}}^{(i+1)}],
\end{aligned}$$

where $\eta_t^{(i)}$ is the number of attributes in each group $t$ at level $i$.

One can observe that the grouping mechanism expressed in Eq. (10) can be alternatively interpreted as a shuffling mechanism written in matrix form denoted as $\boldsymbol{Q}^{(i)}$. Therefore, the transformation from step $i \rightarrow i+1$ is achieved as

$$\boldsymbol{X}^{(i+1)} = \boldsymbol{Q}^{(i)}\boldsymbol{X}^{(i)}\boldsymbol{P}^{(i)}. \tag{10}$$

The two stages of the proposed procedure are continued till the stopping criterion is satisfied. At this point, the dimension-reduction procedure is complete. For the example considered here, the dimension-reduction process is continued for two steps.

Depending on the number of attributes desired by the practitioner or the criterion on information loss, the proposed process of transformation and corresponding group-wise organization can be continued for $I$ steps, resulting in $\boldsymbol{X}^{(I)}$. Then, a generalization for these $I$ transformations can be written as

$$\boldsymbol{X}^{(I)} = \quad \boldsymbol{Q}^{(I)} \cdots \boldsymbol{Q}^{(1)}\boldsymbol{X}^{(1)}\boldsymbol{P}^{(1)} \cdots \boldsymbol{P}^{(I)}, = \boldsymbol{Q}\boldsymbol{X}^{(1)}\boldsymbol{P},$$

where $\boldsymbol{Q} = \boldsymbol{Q}^{(I)} \cdots \boldsymbol{Q}^{(1)}$ and $\boldsymbol{P} = \boldsymbol{P}^{(1)} \cdots \boldsymbol{P}^{(I)}$.

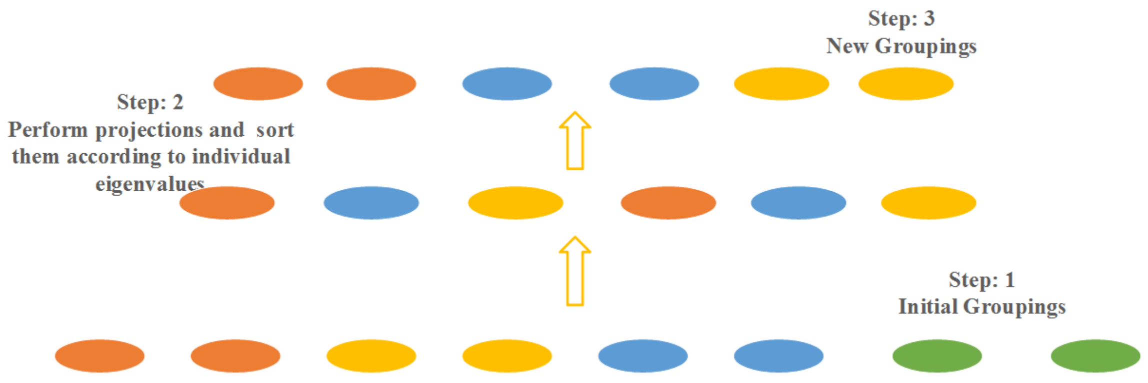

Step: 3
New Groupings
Step: 2
Perform projections and sort them according to individual eigenvalues
Step: 1
Initial Groupings

Figure 3. Overview of the dimension-reduction methodology

**Remark 15** *1. By using the proposed strategy and by controlling the size of the group, one can ensure that dependencies can be evaluated efficiently even in situations when the number of dimensions far exceeds the number of observations as shown here. Thus, this estimation process can be very effective in high-dimensional cases.*

*2. When batch-wise updates are used to define the group-wise approximation as performed here, there is a danger of the newer batch changing $\kappa$ in every group. Due to this, the group-wise organization from that step forward might change, leading to an incorrect estimation. The reason for this is the possibility that the batches till now did not give an adequate estimation of the dependency matrix or that the newer batch represents noisy samples. To avoid complications due to this, the group sizes are fixed after the first batch. The situation can also be avoided by increasing the size of the first batch and/or hand-picking the first batch such that it is optimal. This update process is continuously performed until all the batches are used for training. The whole training process is repeated until convergence or for a pre-defined number of iterations.*

The procedure explained above can be applied on all the observations in the test sample to obtain the dimension reduced data and the step-by-step details for the proposed dimension-reduction are described in Algorithm 3. Let the transformation be denoted as

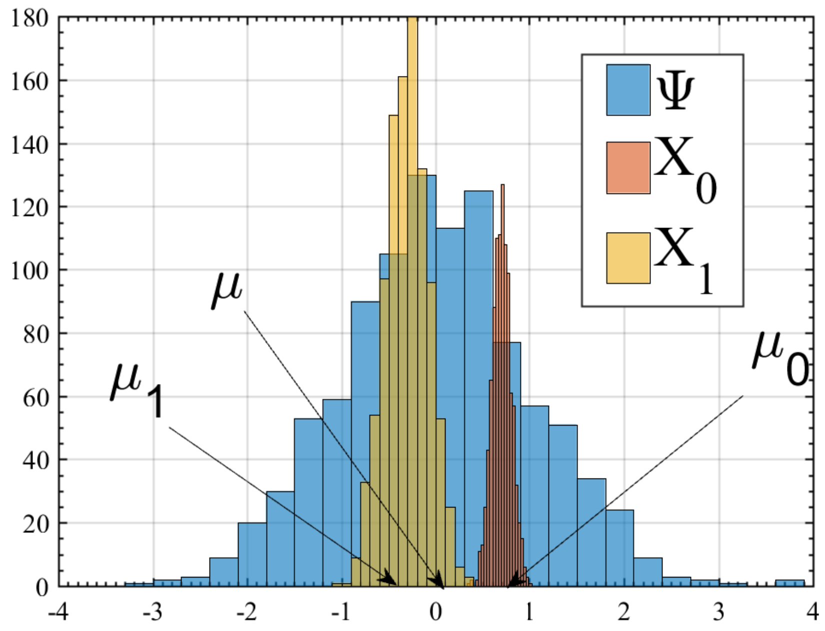


Figure 4. Distribution of $\psi$, $\boldsymbol{X}$ and $\boldsymbol{X}_B$

$\boldsymbol{X}^{(I)} = \phi(\boldsymbol{X}^{(1)})$. Any test datapoints $\boldsymbol{y}$ can be transformed by applying $\phi$ on $\boldsymbol{y}$

$$\boldsymbol{y}^{(I)} = \phi(\boldsymbol{y}). \tag{11}$$

In NDR, all datapoints in the data-set were assumed to be available. In a big-data environment, data is distributed, which would make the native approach for evaluating dependencies practically infeasible. Furthermore, as the dependence relationships evolves across different data samples, statistical heterogeneity would be observed [6]. To understand this effect, consider a system where big data is collected from one attribute and is stored at multiple locations. Let the underlying distribution be represented by $\Psi$ with mean $\mu$, as shown in Figure 4.

Consider two data samples from the underlying distribution as $\boldsymbol{X}_0$ and $\boldsymbol{X}_1$ with means $\mu_0$ and $\mu_1$, respectively. Neither of the two samples represent $\Psi$ fully because there are certain characteristics of $\Psi$ not covered by the data, as shown in Figure 4. As a result, methodologies that learn to classify with $\boldsymbol{X}_0$ or $\boldsymbol{X}_1$ will not capture complete information about $\Psi$. One way to mitigate this problem is to aggregate the information derived from multiple samples.

## 3.2. BATCH-WISE ESTIMATION OF DC AND SVD

For illustration, consider $\boldsymbol{\mathcal{X}}$ with $p = 2$ and let $\boldsymbol{\mathcal{X}}$ be available in two batches denoted as $\boldsymbol{X}_0$ and $\boldsymbol{X}_1$ such that $\boldsymbol{X}_0 = [\boldsymbol{a} \quad \boldsymbol{b}]$ and $\boldsymbol{X}_1 = [\boldsymbol{a}_N \quad \boldsymbol{b}_N]$, where $\boldsymbol{a}$ and $\boldsymbol{b}$ are vectors of length $n$ and $\boldsymbol{a}_N, \boldsymbol{b}_N$ are vectors of length $N$. The objective is to update $v_n^{\boldsymbol{a},\boldsymbol{b}}$ estimated from $\boldsymbol{X}_0$ using Eq. (5) with information from $\boldsymbol{X}_1$.

Therefore, define two augmented vectors $\hat{\boldsymbol{a}} = \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{a}_N \end{bmatrix}$ and $\hat{\boldsymbol{b}} = \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{b}_N \end{bmatrix}$ such that the length of vectors $\hat{\boldsymbol{a}}$ and $\hat{\boldsymbol{b}}$ is now $(n + N)$. The update for distance covariance and/or correlation is given by the following lemma.

**Lemma 3** *Define two augmented data matrices $\hat{\boldsymbol{a}}$ and $\hat{\boldsymbol{b}}$ and let $\boldsymbol{a}_N$ and $\boldsymbol{b}_N$ be the new samples for updating DC. The update for distance covariance is then given as*

$$v^{\hat{\boldsymbol{a}},\hat{\boldsymbol{b}}} = \frac{1}{n^2} v^{\boldsymbol{a},\boldsymbol{b}} + 2\frac{1}{(n+N)^2} v^{\hat{\boldsymbol{a}}\times\hat{\boldsymbol{b}}} + \frac{1}{N^2} v^{\boldsymbol{a}_N,\boldsymbol{b}_N}, \tag{12}$$

*where $v_{\boldsymbol{a},\boldsymbol{b}}$ is the old estimate of DC with distance correlation estimate given as*

$$r^{\hat{\boldsymbol{a}},\hat{\boldsymbol{b}}} = \frac{1}{M^2} r^{\boldsymbol{a},\boldsymbol{b}} + 2\frac{1}{(M+N)^2} r^{\hat{\boldsymbol{a}}\times\hat{\boldsymbol{b}}} + \frac{1}{N^2} r^{\boldsymbol{a}_N,\boldsymbol{b}_N},$$

*where $r_{\boldsymbol{a},\boldsymbol{b}}$ is the old estimate of distance correlation.*

*Proof: See Appendix*

The first term in Eq. (12) represents the distance covariance that is evaluated from $\boldsymbol{X}_0$. The second term is the distance covariance value between the samples $\boldsymbol{a}, \boldsymbol{b}$ and the new batch $\boldsymbol{a}_N, \boldsymbol{b}_N$. The cross term can also be understood as the measure of change in relationships between the current sample and the new batch. The final term in Eq. (12) refers to the distance covariance estimated from the new batch of data. In a similar line of thought, distance covariance/correlation matrix for the multi-variate scenario can be written as

$$\boldsymbol{C}^{\hat{\boldsymbol{a}},\hat{\boldsymbol{b}}} = \boldsymbol{C}^{\boldsymbol{a},\boldsymbol{b}} + 2\boldsymbol{C}^{\hat{\boldsymbol{a}}\times\hat{\boldsymbol{b}}} + \boldsymbol{C}^{\boldsymbol{a}_N,\boldsymbol{b}_N}.$$

Information about nonlinear relationships within any set of attributes is provided by the distance correlation matrices. Using the above proposed procedure, an aggregation of distance correlation can be performed. The expression for aggregating the SVD is presented by the following lemma.

**Lemma 4** *Let the SVD of $\boldsymbol{C}_{\boldsymbol{a},\boldsymbol{b}}$ be given as $\boldsymbol{U\Sigma V}$ and let $\boldsymbol{C}^{\hat{\boldsymbol{a}}\times\hat{\boldsymbol{b}}}, \boldsymbol{C}^{\boldsymbol{a}_N,\boldsymbol{b}_N}$ be evaluated from the new batch of data. The overall SVD of $\boldsymbol{C}_{\boldsymbol{a},\boldsymbol{b}}$ is given as $\boldsymbol{C}_{\hat{\boldsymbol{a}},\hat{\boldsymbol{b}}} = [\boldsymbol{U} \quad I]\hat{\boldsymbol{U}}\hat{\boldsymbol{\Sigma}}\hat{\boldsymbol{V}}^T[\boldsymbol{V} \quad \boldsymbol{J}]^T$, where $\boldsymbol{K} = \begin{bmatrix} \boldsymbol{\Sigma} & 0 \\ 0 & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{V}^T\boldsymbol{J} \\ 0 & \boldsymbol{R_J} \end{bmatrix}$ and $\hat{\boldsymbol{U}}\hat{\boldsymbol{\Sigma}}\hat{\boldsymbol{V}}^T = SVD(\boldsymbol{K})$, and $\boldsymbol{U}, \boldsymbol{V}$ and $\boldsymbol{\Sigma}$ are previously known SVD where $\boldsymbol{J} = 2\boldsymbol{C}^{\hat{\boldsymbol{a}}\times\hat{\boldsymbol{b}}} + \boldsymbol{C}^{\boldsymbol{a}_N,\boldsymbol{b}_N}$ and $\boldsymbol{R_J} = (\boldsymbol{J} - \boldsymbol{V}\boldsymbol{V}^T\boldsymbol{J})$. Proof: See Appendix*

Similar to Eq. (12), $\boldsymbol{U}, \boldsymbol{V}$ and $\boldsymbol{\Sigma}$ are already known. The information about $\hat{\boldsymbol{U}}, \hat{\boldsymbol{\Sigma}}$ and $\hat{\boldsymbol{V}}^T$ is derived from the new batch of data. Furthermore, the size of $\boldsymbol{C}$ is fixed. It is possible that, in some cases, numerical instability is observed during estimation. This problem can be easily fixed using Gram-Schmidt orthogonalization procedures [26].

For practical implementations, the batch-wise updates defined in Lemma 3 and 4 are slightly modified. To this end, consider the data-set be divided into $M$ batches such that $n = n_1 + n_2 \cdots + n_M$. Further assume that $b_q$ represents the batch $q$, with $q$ being the index. Considering this, rewrite the batch-wise updates as

$$\boldsymbol{C}^{(q)} = \boldsymbol{C}^{(q-1)} + 2\hat{\boldsymbol{C}}^{(q-1)\times(q)} + \hat{\boldsymbol{C}}^{(q),(q)}, \tag{13}$$

where $\boldsymbol{C}^{(q)}$ represents the aggregated estimate of distance covariance up to and including batch $b_{q-1}$. The factor $\hat{\boldsymbol{C}}^{(q-1)\times(q)}$ is the estimate with the current batch of data (that is batch $b_q$), and the previously aggregated batch estimate. Finally, $\hat{\boldsymbol{C}}^{(q),(q)}$ is the estimate involving the current batch $b_q$. Similarly, the updates for the singular value decompositions are given for each iteration as

$$\begin{aligned} \boldsymbol{U}^{(q)} &= [\boldsymbol{U}^{(q-1)} \quad I]\hat{\boldsymbol{U}}^{(q)} \\ \boldsymbol{V}^{(q)} &= [\boldsymbol{V}^{(q-1)} \quad I]\hat{\boldsymbol{V}}^{(q)} \\ \boldsymbol{\Sigma}^{(q)} &= \hat{\boldsymbol{\Sigma}}^{(q)}, \end{aligned} \tag{14}$$

where $\hat{\boldsymbol{U}}^{(q)}\hat{\boldsymbol{\Sigma}}^{(q)}(\hat{\boldsymbol{V}}^{(q)})^T = SVD(\boldsymbol{K})$ and $\boldsymbol{K} = \begin{bmatrix} \hat{\boldsymbol{\Sigma}}^{(q)} & \hat{\boldsymbol{\Sigma}}^{(q)}(\hat{\boldsymbol{V}}^{(q-1)})^T\boldsymbol{J}^{(q)} \\ 0 & \boldsymbol{R}_{\boldsymbol{J}}^{(q)} \end{bmatrix}$.

Furthermore, $\hat{\boldsymbol{U}}^{(q)}, \hat{\boldsymbol{\Sigma}}^{(q)}$ and $\hat{\boldsymbol{V}}^{(q)}$ are decomposition from $\boldsymbol{K}^{(q)}$ with $\boldsymbol{J}^{(q)} = 2\hat{\boldsymbol{C}}^{(q-1),(q)} + \hat{\boldsymbol{C}}^{(q),(q)}$ and $\boldsymbol{R}_{\boldsymbol{J}}^{(q)} = (\boldsymbol{J}^{(q)} - \hat{\boldsymbol{V}}^{(q-1)}(\hat{\boldsymbol{V}}^{(q-1)})^T\boldsymbol{J}^{(q)})$. In practical scenarios, Eq (14), (13) can be used to update DC and its corresponding SVD when new data is observed.

**Remark 16** *It can be observed that to estimate the middle term in the batch-wise updates in Eq* (13) *and* (14)*, one needs to know the data corresponding to the previous batches.*

*Theoretically, it can be shown that if an infinite number of datapoints are available then the estimated DC and its SVD will converge to the true values. This result is presented in the appendix; see Theorem 1.*

*However, in practical cases, it is sufficient to maintain a representative sample of the batches seen till now to enable these updates. This history could be generated using popular sampling techniques.*

With the proposed update mechanism, the dimension-reduction method can explore the complete distribution of data while evaluating the transformation parameters. Furthermore, parallelized and distributed estimation is possible using the proposed updates. In [30], a methodology using Hadoop's distributed file system (HDFS) and Apache Spark is introduced for implementing PCA. Similar to [30], NDR can be implemented on such a framework where the original data-set may be stored in HDFS. For each partition of data, the batch wise estimation procedure presented in this paper can be used to evaluate the dimension reducing transformation. Furthermore, it is also possible to implement the group-wise estimation procedures on separate CPU cores.

In these scenarios, the computational overhead for NDR is influenced by the number of groups and the size of the batch. The following lemma attempts to quantify the computational overhead of NDR.

**Lemma 5** *The worst case computational complexity of calculating NDR is in the order of*

$$p\left(\frac{(\eta^I - 1)}{(\eta - 1)\eta^I}\right)O[\eta^3 + MlogM],$$

*where $\eta$ is the number of attributes in a group and $M$ is the size of the batch. Proof: See Appendix*

Observe that the computational complexity of the proposed methodology is a function of both the group-size and the total number of samples in a batch. By keeping both of these quantities small, one is able keep the computational overload to a minimum. However small group size increases the total amount of information that is ignored while capturing relationships. On the other hand, using a large group size can minimize this information loss. Similarly, a large batch size would allow estimation of precise sample parameters, whereas small batch sizes would lead to estimation error. By carefully choosing the batch size and the group sizes for the application at hand, one can manage this trade-off. It is therefore our recommendation that the group size should always be smaller than the batch size for efficiency; otherwise the $p << n$ problem would be observed.

**Algorithm 3** Training for regression parameter estimation

```
1:  Input: Training batch.
2:  for step = 1 $\rightarrow$ Number of Leap Frog updates. do
3:      A sample from the batch
4:      Evaluate $H$.
5:      Half update of momentum.
6:      Update theta based on momentum.
7:      Half update of momentum.
8:      Evaluate $H_{new}$.
9:      if $H_{new} - H < 0$. then
10:         Accept state change.
11:     else
12:         Keep current $\boldsymbol{\theta}$.
13:     end if
14: end for
```

**Algorithm 4** Low-rank approximation for SVD, $j$ represents the dimension in the matrix, $d$ represents the singular value

```
1:  Inputs: matrix $\boldsymbol{K}$, $\kappa$.
2:  Outputs: $\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}$.
3:  Let $C_1 = C$.
4:  Let $\kappa_1 = 0$.
5:  for $j \in 1, 2, ..., \kappa$. do
6:      for Iterate until convergence, $\forall k$. do
7:          $\boldsymbol{u}_k \leftarrow \frac{(\boldsymbol{K}^T\boldsymbol{K})^k \boldsymbol{u}_0}{\|(\boldsymbol{K}^T\boldsymbol{K})^k \boldsymbol{u}_0\|}$
8:          $\boldsymbol{u}_k \leftarrow \frac{(\boldsymbol{K}^T\boldsymbol{K})^k \boldsymbol{v}_0}{\|(\boldsymbol{K}^T\boldsymbol{K})^k \boldsymbol{v}_0\|}$.
9:      end for
10:     $d_j \leftarrow \boldsymbol{u}_j^T \boldsymbol{C}_j \boldsymbol{v}_j$
11:     $\boldsymbol{C}_j = \boldsymbol{C}_{j-1} - d_j \boldsymbol{u}_j^T \boldsymbol{v}_j$.
12: end for
13: Final SVD, $\boldsymbol{U} = \{\boldsymbol{u}_j\}, \boldsymbol{V} = \{\boldsymbol{v}_j\}, \boldsymbol{\Sigma} = diag(\{d_j\}), \forall j = 1, 2, \cdots, \kappa$.
```

**Algorithm 5** A Nonlinear dimension-reduction (NDR)

```
1:  Input: $\mathcal{X}$, $\alpha$
2:  Output:Reduced dimension $\mathcal{X}$
3:
4:  for Each $batch$ in $\mathcal{X}$ do
5:      $\boldsymbol{X}^{(1)} = batch$
6:      Standardize $\boldsymbol{X}^{(1)}$.
7:      for step from $i = 1 \rightarrow I$ do
8:          if $\frac{1}{T^{(i)}} \sum_{t=0}^{T^{(i)}} \sum_{l=0}^{\kappa} \lambda_l \geq \alpha$ then
9:              Stop the dimension-reduction procedure
10:         end if
11:         if i>1 then
12:             Create groups
13:         else
14:             Generate groupings at random
15:         end if
16:         for Every group at step $i$ do
17:             Calculate Distance Correlation matrix
18:             if batch number is one then
19:                 Use current Distance Correlation
20:                 Evaluate $\kappa$
21:             else
22:                 Use the aggregated Distance Correlation
23:                 Evaluate $\kappa$
24:             end if
25:             Evaluate low rank approximation using Alg. 4
26:             Update distance covariance using Eq (13)
27:             Update low rank SVD using Eq (14).
28:         end for
29:     end for
30:     Train the regression parameter using Alg. 5
31: end for
```

As described in Eq. (2), the objective was to determine $p(\boldsymbol{y} \in \boldsymbol{\Psi}_k|\boldsymbol{X})$. Due to the dimension-reduction transformation $\phi$, one may rewrite Eq. (2) as

$$p(\boldsymbol{y} \in \boldsymbol{\Psi}_k|\boldsymbol{X}) = f(\phi(\boldsymbol{y}); \boldsymbol{\theta}), \tag{15}$$

where $f$ represents the regression function and $\boldsymbol{\theta}$ represents the parameters. A total of $\mathcal{F}$ probabilities are evaluated using Eq. (15). Finally, the probability with the maximum value determines the state of the system using $\boldsymbol{y}$ which is given as

$$p_k^* = \arg\max\{p(\phi(\boldsymbol{y}) \in \boldsymbol{\Psi}_k|\boldsymbol{X}), k = 0, 1, \cdots, \mathcal{F}\}. \tag{16}$$

The index with the winning probability $p_k^*$ indicates the class of the sample or the corresponding fault.

To learn the regression coefficients, Hamiltonian (cost function) $H$ is first defined as described in [31] and the optimization problem is given as

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} H(\boldsymbol{\theta}), \tag{17}$$

where $H(\boldsymbol{\theta})$ is an appropriate cost function and cross-entropy is the most popular choice [9].

To optimize Eq (17), Hamiltonian Monte Carlo-based leap frog optimization (HMC) is performed [31].First, a small sample of data is extracted from $\boldsymbol{X}^{(I)}$ using Monte Carlo sampling and the gradient of $H(\boldsymbol{\theta})$ is evaluated using the sample. Next, momentum $\boldsymbol{v}$ is calculated to get

$$\boldsymbol{v} = \boldsymbol{v} - \epsilon\frac{\Delta_{\boldsymbol{\theta}} H(\boldsymbol{\theta})}{2}. \tag{18}$$

The parameters are then updated based on momentum, given as

$$\boldsymbol{\theta} = \boldsymbol{\theta} + \epsilon\boldsymbol{v}, \tag{19}$$

where $\epsilon$ is the step size and $\Delta_{\boldsymbol{\theta}} H(\boldsymbol{\theta})$ denotes the gradient of the cost function with respect to $\boldsymbol{\theta}$. Once the new parameters are obtained, the cost is evaluated to see if the update must be allowed or not. For this, the Hamiltonian value (cost function) is calculated again. The transition is allowed if and only if

$$dH < 0. \tag{20}$$

Otherwise, resampling is performed. The overall algorithm for the optimization is described in Alg. 3.

The application of the proposed methodology on real-world and synthetic data-sets is presented next.

## 4. RESULTS AND DISCUSSIONS

A total of ten data-sets are used for analysis in this study. The statistics for these data-sets are tabulated in Table 2. Each data-set is randomly split into test and train data. The split is chosen as 20% for test and 80% for training.

The fault diagnostics performance of the methodology will be shown first using the rolling element bearing data-set and the sensorless drive diagnostics data-set. Next, the classification performance of the NDR is shown on the rest of the data-sets that are tabulated in Table 2. All results in this section are averaged over 1000 iterations. Hyper parameters for all of the classification methodologies are tabulated in Table 3.

### 4.1. FAULT DIAGNOSTICS

**4.1.1. Rolling Element Bearing Data-set.** There are eleven attributes in the data. Three types of faults are under analysis. One thousand data points are randomly sampled from the data-set to constitute the healthy sample and each of the three fault conditions. The choice of the number 1000 for the preprocessing data is arbitrary.

Table 2. Summary descriptions of the different data-sets used in this paper

| **Data-set** | **Dimensions** | **Data points** | **Classes** |
|---|---|---|---|
| Rolling [32] | 11 | 35000 | 4 |
| Sensorless [33] | 48 | 78000 | 11 |
| MNIST [5] | 784 | 72000 | 10 |
| NotMNIST [34] | 784 | 81000 | 10 |
| CiFAR-10 [4] | 3072 | 50000 | 10 |
| Gisette [35] | 5000 | 6000 | 2 |
| Madelon [35] | 500 | 1000 | 2 |
| Arcene [35] | 10000 | 100 | 2 |
| Dexter [35] | 20000 | 300 | 2 |
| Image-Net [36] | 784 | 12614060 | 10 |
| Synthetic Data-set | 20000 | 2000000 | 10 |

Table 3. Hyper-parameters for different methodologies in this paper.

| **Method** | **Hyper-parameters** |
|---|---|
| KNearest Neighbors (KNN) | Three neighbors |
| Support Vector Machine (SVM) | C = 0.025 |
| Kernel SVM | RBF kernel, gamma = 2, C=1 |
| Desicion Trees (DT) | Max depth =5 |
| Random Forest (RF) | Max depth=5 estimators=10, max attributes=1 |
| Shallow Neural Network (SNN) | Lr = 0.01, 1 hidden layer, 500 neurons |
| AdaBoost | Number of estimators=50, learning rate=1.0 |
| Naive-Bayes | No Priors |
| Quadratic Discriminant Analysis (QDA) | No Priors |
| Logistic Regression (LR) | Random initialization, learning rate = 0.001, 1000 iterations |

Table 4. Accuracies at various dimensions along with false positive rates for rolling element bearing data-set.

| $\alpha/100$ | **Dimensions** | **False Positive Rates** | **Accuracy** |
|---|---|---|---|
| 0.50 | 5 | 0.008 | 0.93 |
| 0.60 | 6 | 0.009 | 0.94 |
| 0.70 | 7 | 0.011 | 0. 96 |
| 0.95 | 10 | 0.011 | 0.98 |
| 0.99 | 11 | 0.011 | 0.95 |

To test the performance of NDR, false positives and accuracies with respect to the number of dimensions extracted from the data are listed in Table 4. With a decrease in the total number of dimensions derived from the data, the false positive rate decreases. The case

Table 5. Accuracy for the proposed methodology at various dimensions for the sensorless drive diagnostics data-set.

| $\alpha/100$ | **Dimensions** | **False Positive Rates** | **Accuracy** |
|---|---|---|---|
| 0.7 | 20 | 0.008 | 0.91 |
| 0.75 | 25 | 0.009 | 0.91 |
| 0.80 | 30 | 0.011 | 0. 81 |
| 0.85 | 35 | 0.031 | 0.78 |
| 0.95 | 40 | 0.023 | 0.76 |
| 0.99 | 48 | 0.021 | 0.77 |

Table 6. Average classification accuracy (Avg. Class. Acc.) rates for various dimensions reduction techniques (Dim-Red) and classifiers using the Sensorless Drive Diagnostics data-set.

| **Dim-Red ↓** | **KNN** | SVM | LDA | DT | RF | SNN | AdaBoost | Naive-Bayes | QDA |
|---|---|---|---|---|---|---|---|---|---|
| PCA | 0.52 | 0.52 | 0.52 | 0.45 | 0.45 | 0.52 | 0.18 | 0.45 | 0.53 |
| ISOMAP | 0.67 | 0.46 | 0.43 | 0.32 | 0.39 | 0.58 | 0.18 | 0.42 | 0.49 |
| LLE | 0.25 | 0.08 | 0.15 | 0.18 | 0.18 | 0.08 | 0.18 | 0.19 | 0.19 |
| KPCA | 0.54 | 0.53 | 0.52 | 0.45 | 0.44 | **0.58** | 0.28 | 0.45 | 0.53 |
| **NDR** | **0.89** | **0.91** | **0.60** | **0.50** | **0.95** | 0.47 | **0.74** | **0.83** | **0.84** |

with 10 dimensions provides better accuracy compared to no dimension reduction. This can lead to a conclusion that the data-set has a particular dimension, which is redundant. When the value of $\alpha/100$ is set to be 0.95, NDR can only reduce dimensions to 10. By reducing the value of $\alpha$, further reduction can be achieved. Also note that the case with 5 dimensions is accomplished with $\alpha/100$ of 0.5.

Different classification methodologies are used in conjunction with NDR. The results are shown in the first row of Table 7 and the results indicate the efficiency of NDR to capture information in the data. The rolling element data-set is a benchmark data-set for the proposed methodology. Here, a traditional dimension-reduction approach provides reasonable results. In the next data-set, some of the traditional methodologies fail due to the presence of complicated relationships, as will be shown.

Table 7. Accuracies for the various data-sets with different methods using NDR as the dimension-reduction methodology with generalization error (the difference between the test and the training error) in parenthesis. The value of $\alpha$ is chosen as 0.95 for all the data-sets.

| | **KNN** | **SVM** | **LDA** | **DT** | **RF** | **SNN** | **AdaB** | **LR** |
|---|---|---|---|---|---|---|---|---|
| **Rolling** | 0.99 (0.006) | 0.96 (0.017) | 0.96 (0.004) | 0.95 (0.019) | 0.97 (0.0077) | 0.88 (0.014) | 0.86 (0.034) | 0.97 (0.019) |
| **Sensorless** | 0.88 (0.008) | 0.91 (0.01) | 0.60 (0.016) | 0.50 (0.013 ) | 0.95 (0.021) | 0.47 (0.004) | 0.74 (0) | 0.93 (0.004) |
| **MNIST** | 0.91 (0.031) | 0.91 (0.006) | 0.82 (0.024) | 0.64 (0.0836) | 0.62 (0.021) | 0.96 (0.024) | 0.721 (0.039) | 0.96 (0.083) |
| **NotMnist** | 0.90 (0.044) | 0.82 (0.016) | 0.88 (0.033) | 0.82 (0.022) | 0.66 (0.026) | 0.93 (0.014) | 0.72 (0.025) | 0.94 (0.014) |
| **CiFAR-10** | 0.29 (0.080) | 0.44 (0.047) | 0.37 (0.056) | 0.26 (0.048) | 0.21 (0.090) | 0.50 (0.036) | 0.51 (0.077) | 0.66 (0.036) |
| **Madelon** | 0.49 (0.28) | 0.57 (0.21) | 0.58 (0.72) | 0.56 (0.081) | 0.54 (0.183) | 0.53 (0.201) | 0.59 (0.195) | 0.61 (0.081) |
| **Dexter** | 0.48 (0) | 0.54 (0) | 0.54 (0) | 0.77 (0) | 0.53 (0) | 0.57 (0) | 0.69 (0) | 0.81 (0.015) |
| **Gisette** | 0.94 (0.031) | 0.93 (0.012) | 0.91 (0.052) | 0.92 (0.049) | 0.74 (0.074) | 0.98 (0.015) | 0.97 (0.019) | 0.99 (0.015) |

### 4.1.2. Sensorless Drive-Diagnostics Data-set.

Sensorless Drive Diagnostics data-set is significantly larger than rolling element bearingdata-set. The problem considered here is an 11-fault estimation problem involving 48 dimensions. A total of 1000 observations belonging to every fault are sampled and NDR is used to reduce dimensions in the data. The resulting accuracies on the test-set are shown in the second row of Table 7.

As seen in Table 7, the detection results are optimal. Large values of accuracies and low values of false positive error rates indicate this optimality. In this data-set, it is observed that the Type-1 error rate increases with an increase in the number of dimensions. The best-case accuracy is achieved in this case when the total number of dimensions is reduced to 35.

Based on the results in Table 7, NDR is able to capture information accurately. Note that the results for DT, RF, Ada-boost, and Naive-Bayes are suboptimal for this data-set. For further investigation, these methods are applied on the sensorless drive diagnostics data-set without dimension reduction. It was confirmed from the results that these methods are unsuitable for the sensorless drive data-set.

Next, NDR is compared to the state-of-the-art dimension reduction techniques. The results are summarized in Table 6[1]. The use of NDR results in better accuracies compared to the rest of the dimension-reduction methods.

Since the sensorless drive data-set contains nonlinear relationships, PCA is expected to fail, and this is observed from the results. Furthermore, nonlinear dimension-reduction approaches such as ISOMAP, LLE, and KPCA fail due to the stringency of approximation. In all of these tests, the data are reduced from 48 dimensions to 25. The observation that DT, RF, Adaboost, and Naive-Bayes are suboptimal for this data-set is consistently seen here for various dimension-reduction schemes.

In the above two data-sets, the performance of the overall approach has been established, even for cases where the traditional dimension-reduction approach does not perform well. In the next section, relative performance of NDR on classification of big data is studied and a total of 10 data-sets were used.

## 4.2. CLASSIFICATION

All of the data-sets mentioned in Table 2 are analyzed using NDR in conjunction with 10 popular classification methods on these data-sets. These results are shown in Table 7. It is seen that the NDR performs reasonably well for data-sets with both linear and nonlinear relationships.

To analyze how much information is captured by NDR relative to a standard dimension-reduction method such as PCA, a linear regression approach is utilized. Each PCA component is regressed against components derived from NDR. The goodness of fit for these linear regressions is measured by averaging the $R^2$ value for varying number of PCA components for an equal number of NDR components. For example, each of the first five components from PCA are regressed against the first five components from NDR, and

[1]Acronyms for different dimension-reduction techniques are PCA (Principal Component Analysis), ISOMAP (Isometric Mapping), FastICA (Independent Component Analysis), LLE (Locally Linear Embedding), and NDR (Nonlinear Dimension-reduction) [37].

then the $R^2$ values are averaged for each execution. This test is executed 1000 times and $5, 10, 15, 20$ components are compared. Large $R^2$ averages indicate that we can reconstruct the principal components relatively accurately from NDR components as seen in Table 8. The results suggest that the multi-step NDR approach does not result in appreciable information loss relative to the standard PCA method since the $R^2$ averages are large. Note that for this test, synthetic data are generated with varying number of observations and a total of 100 dimensions.

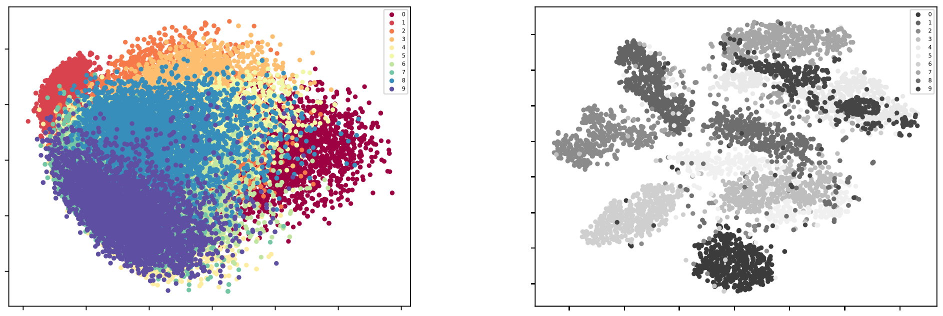
Finally, NDR is tested on a very large data-set consisting of 20,000 dimensions and 2,00,00,00 datapoints. While generating this data-set, clusters of normally distributed points are created about vertices of a 20,000-dimensional hypercube with sides of a predefined length which is chosen as one for this case. An equal number of clusters is constructed for each class. Note that the datapoints are equally distributed among all the classes in consideration. A python function for generating this data-set is available in the standard sklearn python library [38]. The ten classes in the data are visualized using NDR and the results are compared with TSNE. Therefore, five thousand datapoints corresponding to each class are sampled and two dimensions are extracted using both NDR and TSNE.

The results in Figure 5 highlight the application of the proposed methodology to visualization. The results indicate that NDR is able to reasonably disentangle nonlinearities in the synthetic data-set as seen from Figure 5a and can act as a useful tool in the domain of data visualization. In comparison, TSNE provides clear separation between different classes, as seen in Figure 5b. However, it must be observed that the computational load of using NDR is eighty times lower than that of TSNE, which is observed in Table 9. Furthermore, the increased computational load comes at the price of 2% improvement in accuracy, as seen in Table 9, when LR is used for obtaining the result.

The main idea behind the proposed methodology is to group the attributes and reduce the dimensions in each group. As a result, each group at a particular step can be implemented independently. Using the python multi-processing library, NDR is implemented on

Table 8. Approximate $R^2$ values between the components derived from PCA compared to components derived from NDR for an increasing number of datapoints.

| Number of Components | 4096 | 2048 | 1024 | 56 | 10 | 5 |
|---|---|---|---|---|---|---|
| 5 | 0.8516 | 0.8615 | 0.8975 | 0.8328 | 0.9569 | 1. |
| 10 | 0.8895 | 0.8826 | 0.8554 | 0.9014 | 1. | 1. |
| 15 | 0.8995 | 0.9026 | 0.9554 | 0.9901 | 1. | 1. |
| 20 | 0.8921 | 0.9014 | 0.9210 | 0.9964 | 1. | 1. |



(a) (b)

Figure 5. Visualization results for the (a) NDR (b) TSNE.

Table 9. Average accuracies with computational times for the synthetic data-set.

| Methodology | Accuracy | Computational Times (sec) |
|---|---|---|
| NDR | 96.10 | 57 |
| TSNE | 98.41 | 4834 |

Table 10. Average accuracies with computational times for parallelized implementation on a twelve core CPU.

| Data-set | Accuracy | Computational Times (sec) |
|---|---|---|
| Synthetic | 94.10 | 377 |
| Image-Net | 56.21 | 6874 |

a multi-core CPU. Finally, the results are demonstrated on two large data-sets where the first is the popular ImageNet data-set [36], and the second is the synthetic data-set. We construct a ten class problem from the ImageNet data-set by sampling with replacement. In the end, the data-set is about 138 GB in size consisting of ten classes. Both of these data-sets are extremely large and cannot be implemented on a single computer node. The results indicate the ability of the methodology to obtain reasonable performance in parallelized environments. Note that ImageNet is a very complicated data-set and a simple Logistic Regression based classification is not enough for optimum performance. The results demonstrated here indicate the possibility of a parallelized implementation for NDR and do not imply state of the art performance.

These results indicate impressive performance of NDR even in scenarios where the traditional methods do not work very well. Moreover, we note that in data-sets, where the number of observations are fewer than the number of dimensions in the data, the proposed methodology is optimal.

## 5. CONCLUSIONS

In this paper, an iterative dimension-reduction approach is proposed for handling big-data challenges such as nonlinear relationships (NDR), spurious correlations, redundant dimensions, and statistical heterogeneity. The proposed methodology was applied to the problem of fault diagnostics and classification.

NDR achieves high accuracies in the case of large $p$, small $n$ data-sets. Due to the incremental updates proposed in this paper, NDR can address heterogeneity while being effective in large-scale distributed storage environments typical with large $p$, large $n$ scenarios. As information loss is controlled by a user-defined parameter in this approach, NDR can dynamically determine the number of dimensions to be extracted from the data. As a consequence, NDR is able to remove redundant and unwanted dimensions while taking into account spurious correlations efficiently. Due to the use of distance covariance, NDR is also able to capture nonlinear relationships from the data in contrast with standard dimension-reduction approaches such as PCA, FA.

Since, NDR does not take into account the error in the detection process while reducing dimensions, it suffers from ineffectiveness in very complicated data-sets, such as CiFAR-10. Enabling such a feedback loop would be beneficial for improvements in big-data cases and is therefore part of our future work.

## ACKNOWLEDGMENT

The authors would like to thank Dr. John Singler, Associate Professor, Department of Mathematics and Statistics, for his help in verifying the math related to the batch-wise update of the SVD and the related results and Dr. Elizabeth Bowles, Research Associate, Biological Sciences Department, for her help in improving the written flow. This research was supported in part by NSF I/UCRC award IIP 1134721 and Intelligent Systems Center.

## APPENDIX

**0.0.0.1. Proof:.** [Incremental updates for distance covariance] Consider a double centered matrix of pairwise distance $\boldsymbol{A}$ as shown in Eq 4 in the main body of the paper. The updated distance matrix $\hat{\boldsymbol{A}}$ can be written as

$$\hat{\boldsymbol{A}} = \begin{bmatrix} \tilde{\boldsymbol{A}} & \tilde{\boldsymbol{A}}_{\boldsymbol{a},\boldsymbol{a}_N} \\ \tilde{\boldsymbol{A}}_{\boldsymbol{a},\boldsymbol{a}_N} & \tilde{\boldsymbol{A}}_{\boldsymbol{a}_N,\boldsymbol{a}_N} \end{bmatrix}, \tag{21}$$

where $\tilde{\boldsymbol{A}}$ is the old distance matrix with $\tilde{\boldsymbol{A}}_{\boldsymbol{a},\boldsymbol{a}_N}$ representing the distance matrix between old and new samples and $\tilde{\boldsymbol{A}}_{\boldsymbol{a}_N,\boldsymbol{a}_N}$ represents the pairwise distance values within the new sample.

Similarly, consider another augmented distance matrix as $\hat{\boldsymbol{B}}$ given as

$$\hat{\boldsymbol{B}} = \begin{bmatrix} \tilde{\boldsymbol{B}} & \tilde{\boldsymbol{B}}_{\boldsymbol{b},\boldsymbol{b}_N} \\ \tilde{\boldsymbol{B}}_{\boldsymbol{b},\boldsymbol{b}_N} & \tilde{\boldsymbol{B}}_{\boldsymbol{b}_N,\boldsymbol{b}_N} \end{bmatrix}. \tag{22}$$

The squared measure of distance covariance for the augmented matrix can be written in matrix form as

$$\begin{aligned} \nu^{\hat{\boldsymbol{a}},\hat{\boldsymbol{b}}} = [\mathbf{1}_{\boldsymbol{a}}^T \quad \mathbf{1}_{\boldsymbol{a}_N}^T]\left( \begin{bmatrix} \tilde{\boldsymbol{A}} & \tilde{\boldsymbol{A}}_{\boldsymbol{a},\boldsymbol{a}_N} \\ \tilde{\boldsymbol{A}}_{\boldsymbol{a},\boldsymbol{a}_N} & \tilde{\boldsymbol{A}}_{\boldsymbol{a}_N,\boldsymbol{a}_N} \end{bmatrix} \odot \begin{bmatrix} \tilde{\boldsymbol{B}} & \tilde{\boldsymbol{B}}_{\boldsymbol{b},\boldsymbol{b}_N} \\ \tilde{\boldsymbol{B}}_{\boldsymbol{b},\boldsymbol{b}_N} & \tilde{\boldsymbol{B}}_{\boldsymbol{b}_N,\boldsymbol{b}_N} \end{bmatrix} \right) \\ [\mathbf{1}_{\boldsymbol{b}} \quad \mathbf{1}_{\boldsymbol{b}_N}], \end{aligned} \tag{23}$$

where $\odot$ represents the Hadamard product. Simplifying the above expression, one can achieve

$$\nu^{\hat{\boldsymbol{a}},\hat{\boldsymbol{b}}} = [\mathbf{1}_{\boldsymbol{a}}^T \quad \mathbf{1}_{\boldsymbol{a}_N}^T]\left( \begin{bmatrix} \tilde{\boldsymbol{A}} \odot \tilde{\boldsymbol{B}} & \tilde{\boldsymbol{A}}_{\boldsymbol{a},\boldsymbol{a}_N} \odot \tilde{\boldsymbol{B}}_{\boldsymbol{b},\boldsymbol{b}_N} \\ \tilde{\boldsymbol{A}}_{\boldsymbol{a},\boldsymbol{a}_N} \odot \tilde{\boldsymbol{B}}_{\boldsymbol{b},\boldsymbol{b}_N} & \tilde{\boldsymbol{A}}_{\boldsymbol{a}_N,\boldsymbol{a}_N} \odot \tilde{\boldsymbol{B}}_{\boldsymbol{b}_N,\boldsymbol{b}_N} \end{bmatrix} \right) [\mathbf{1}_{\boldsymbol{b}} \quad \mathbf{1}_{\boldsymbol{b}_N}]. \tag{24}$$

After further simplification the final expression can be written as

$$\begin{aligned} \nu^{\hat{\boldsymbol{a}},\hat{\boldsymbol{b}}} = \mathbf{1}_{\boldsymbol{a}}^T(\tilde{\boldsymbol{A}} \odot \tilde{\boldsymbol{B}})\mathbf{1}_{\boldsymbol{b}} + \mathbf{1}_{\boldsymbol{a}}^T(\tilde{\boldsymbol{A}}_{\boldsymbol{a},\boldsymbol{a}_N} \odot \tilde{\boldsymbol{B}}_{\boldsymbol{b},\boldsymbol{b}_N})\mathbf{1}_{\boldsymbol{b}_N} + \mathbf{1}_{\boldsymbol{a}_N}^T(\tilde{\boldsymbol{A}}_{\boldsymbol{a},\boldsymbol{a}_N} \odot \tilde{\boldsymbol{B}}_{\boldsymbol{b},\boldsymbol{b}_N})\mathbf{1}_{\boldsymbol{b}} \\ + \mathbf{1}_{\boldsymbol{a}_N}^T(\tilde{\boldsymbol{A}}_{\boldsymbol{a}_N,\boldsymbol{a}_N} \odot \tilde{\boldsymbol{B}}_{\boldsymbol{b}_N,\boldsymbol{b}_N})\mathbf{1}_{\boldsymbol{b}_N}. \end{aligned} \tag{25}$$

Let $\nu^{\boldsymbol{a},\boldsymbol{b}} = \mathbf{1}_{\boldsymbol{a}}^T(\tilde{\boldsymbol{A}} \odot \tilde{\boldsymbol{B}})\mathbf{1}_{\boldsymbol{b}}$ be the old estimate of the distance covariance. Let $\nu^{\hat{\boldsymbol{a}}\times\hat{\boldsymbol{b}}} = \mathbf{1}_{\boldsymbol{a}_N}^T(\tilde{\boldsymbol{A}}_{\boldsymbol{a},\boldsymbol{a}_N} \odot \tilde{\boldsymbol{B}}_{\boldsymbol{b},\boldsymbol{b}_N})\mathbf{1}_{\boldsymbol{b}}$ be the estimate between the old and new samples with $\nu^{\boldsymbol{a}_N,\boldsymbol{b}_N} = \mathbf{1}_{\boldsymbol{a}_N}^T(\tilde{\boldsymbol{A}}_{\boldsymbol{a}_N,\boldsymbol{a}_N} \odot \tilde{\boldsymbol{B}}_{\boldsymbol{b}_N,\boldsymbol{b}_N})\mathbf{1}_{\boldsymbol{b}_N}$ being the estimate from the new sample alone. Combining all the constant terms, one can achieve

$$\nu^{\hat{\boldsymbol{a}},\hat{\boldsymbol{b}}} = \frac{1}{M^2}\nu^{\boldsymbol{a},\boldsymbol{b}} + 2\frac{1}{(M+N)^2}\nu^{\hat{\boldsymbol{a}}\times\hat{\boldsymbol{b}}} + \frac{1}{N^2}\nu^{\boldsymbol{a}_N,\boldsymbol{b}_N}. \tag{26}$$

Similarly, one can write the correlation estimate as

$$r^{\hat{\boldsymbol{a}},\hat{\boldsymbol{b}}} = r^{\boldsymbol{a},\boldsymbol{b}} + 2r^{\hat{\boldsymbol{a}}\times\hat{\boldsymbol{b}}} + r^{\boldsymbol{a}_N,\boldsymbol{b}_N}, \tag{27}$$

which completes the proof. □

**0.0.0.2. Proof:.** [Incremental update for SVD] Consider

$$\boldsymbol{C}^{\hat{\boldsymbol{a}},\hat{\boldsymbol{b}}} = \boldsymbol{C}^{\boldsymbol{a},\boldsymbol{b}} + 2\boldsymbol{C}^{\hat{\boldsymbol{a}}\times\hat{\boldsymbol{b}}} + \boldsymbol{C}^{\boldsymbol{a}_N,\boldsymbol{b}_N}. \tag{28}$$

Given the SVD of the old estimate of the matrix, merge the two terms on the right as $\boldsymbol{J} = 2\boldsymbol{C}^{\hat{\boldsymbol{a}}\times\hat{\boldsymbol{b}}} + \boldsymbol{C}^{\boldsymbol{a}_N,\boldsymbol{b}_N}$. Split the terms to write

$$\boldsymbol{C}^{\hat{\boldsymbol{a}},\hat{\boldsymbol{b}}} = [\boldsymbol{U} \quad \boldsymbol{I}] \begin{bmatrix} \boldsymbol{\Sigma} & 0 \\ 0 & \boldsymbol{I} \end{bmatrix} [\boldsymbol{V} \quad \boldsymbol{J}]^T, \tag{29}$$

where $\boldsymbol{I}$ is the identity matrix of appropriate dimensions. Next decompose $[\boldsymbol{V} \quad \boldsymbol{J}]$ such that $[\boldsymbol{V} \quad \boldsymbol{J}] = [\boldsymbol{V} \quad \boldsymbol{J} + \boldsymbol{V}\boldsymbol{V}^T\boldsymbol{J} - \boldsymbol{V}\boldsymbol{V}^T\boldsymbol{J}]$ and rearrange to get

$$[\boldsymbol{V} \quad \boldsymbol{J}] = [\boldsymbol{V} \quad \boldsymbol{I}] \begin{bmatrix} \boldsymbol{I} & \boldsymbol{V}^T\boldsymbol{J} \\ 0 & \boldsymbol{R_J} \end{bmatrix}$$

Rewrite the decomposition as

$$\boldsymbol{C}^{\hat{\boldsymbol{a}},\hat{\boldsymbol{b}}} = [\boldsymbol{U} \quad \boldsymbol{I}] \begin{bmatrix} \boldsymbol{\Sigma} & 0 \\ 0 & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} I & \boldsymbol{V}^T\boldsymbol{J} \\ 0 & \boldsymbol{R_J} \end{bmatrix}^T [\boldsymbol{V} \quad \boldsymbol{I}]^T, \tag{30}$$

where $\boldsymbol{R_J} = (\boldsymbol{J} - \boldsymbol{V}\boldsymbol{V}^T\boldsymbol{J})$. Further let $\boldsymbol{K} = \begin{bmatrix} \boldsymbol{\Sigma} & 0 \\ 0 & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{V}^T\boldsymbol{J} \\ 0 & \boldsymbol{R_J} \end{bmatrix}$ to achieve the expression as a product of two orthonormal matrices.

$$\boldsymbol{C}^{\hat{\boldsymbol{a}},\hat{\boldsymbol{b}}} = [\boldsymbol{U} \quad \boldsymbol{I}]\boldsymbol{K}[\boldsymbol{V} \quad \boldsymbol{I}]^T, \tag{31}$$

where $\boldsymbol{K} = \begin{bmatrix} \boldsymbol{\Sigma} & \boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{J} \\ 0 & \boldsymbol{R_J} \end{bmatrix}$. Finally, diagonalize $\boldsymbol{K}$ to achieve the final result of

$$\boldsymbol{C}^{\hat{\boldsymbol{a}},\hat{\boldsymbol{b}}} = [\boldsymbol{U} \quad \boldsymbol{I}]\boldsymbol{U}'\boldsymbol{\Sigma}'\boldsymbol{V}'^T[\boldsymbol{V} \quad \boldsymbol{I}]^T. \tag{32}$$

Since $\boldsymbol{U}$ and $\boldsymbol{V}$ are known and $I$ is the identity matrix. One can diagonalize $\boldsymbol{K}$ to get $\boldsymbol{U}', \boldsymbol{\Sigma}', \boldsymbol{V}'$ and the final incremental updates of SVD. □

**Theorem 2** *Consider the batch-wise update given by Lemma 1 and 2. The updates converge to their true values as $M \longrightarrow \infty$ with $M$ being the number of batches.*

**0.0.0.3. Proof:.** Let the $n$ sample points in the data be divided across $M$ batches and consider the update of the distance given as

$$\boldsymbol{C}^{(q)} = \hat{\boldsymbol{C}}^{(q-1)} + 2\boldsymbol{C}^{(q-1),(q)} + \boldsymbol{C}^{(q),(q)}. \tag{33}$$

where $m$ refers to the batch index . Write the two elements on the right as $\boldsymbol{J}^{(q)} = 2\hat{\boldsymbol{C}}^{(q-1),(q)} + \hat{\boldsymbol{C}}^{(q),(q)}$ to achieve

$$\boldsymbol{C}^{(q)} = \boldsymbol{C}^{(q-1)} + \boldsymbol{J}^{(q)}. \tag{34}$$

As a result the update for the $M^{th}$ batch can be written as

$$\boldsymbol{C}^{(M)} = \boldsymbol{C}^{(M-1)} + \boldsymbol{J}^{(M)}, \tag{35}$$

where $\boldsymbol{C}^{(M-1)}$ is given as

$$\boldsymbol{C}^{(M-1)} = \boldsymbol{C}^{(M-2)} + \boldsymbol{J}^{(M-1)}. \tag{36}$$

Substitution into Eq. (35) reveals

$$\boldsymbol{C}^{(M)} = \boldsymbol{C}^{(M-2)} + \boldsymbol{J}^{(M-1)} + \boldsymbol{J}^{(M)}. \tag{37}$$

Generalizing this for all of the batches and collecting all the terms, the update is written as

$$\boldsymbol{C}^{(M)} = \boldsymbol{C}^{(1)} + \sum_{q=2}^{M-1} \boldsymbol{J}^{(q)}, \tag{38}$$

such that $\boldsymbol{C}^{(M)}$ represents the sample dependency matrix for sample size $n$, where $n$ is the cumulative sum of sample points across all of the batches. It is known that $\boldsymbol{C}^{(M)}$ will converge to its true estimate almost surely as $n \longrightarrow \infty$ [8] and the left side in Eq. (38) converges. Therefore, as $M \longrightarrow \infty$, $\sum_{q=1}^{M-1} \boldsymbol{J}^{(q)}$ converges to its true values which implies that the updates converge. Moreover, since the SVD updates are exact, the convergence follows directly from the fact that $C^{(M)}$ converges which was just shown. □

**0.0.0.4. Proof:.** [Computational Complexity Lemma] At the first step in NDR, the total number of attributes are paired into groups of $\eta$. As a result there are a total $p/\eta$ groups at the first step. Let there be $I$ steps in the dimension-reduction process. The total number of groups upto the penultimate step of the dimension-reduction process assuming equal group sizes can be counted as

$$p \sum_{i=1}^{I} \frac{1}{\eta^i} \left( \frac{(\eta^I - 1)}{(\eta - 1)\eta^I} \right). \tag{39}$$

The most computationally expensive operation is that of calculating the distance covariance and corresponding SVD. This process has a complexity of $O[MlogM]$ and $O[\eta^3]$ [39], respectively, where $M$ is the size of the batch and $\eta$ is the size of the distance covariance matrix. As a result the worst case upper bound for dimension-reduction in one group can be counted as $O[\eta^3 + MlogM]$ [40]. We can therefore write the total approximate asymptotic complexity from the lowest step until the ultimate step for one batch as

$$p(\frac{(\eta^I - 1)}{(\eta - 1)\eta^I})O[\eta^3 + MlogM]. \tag{40}$$

This completes our proof. □

## REFERENCES

[1] C. P. Chen and C.-Y. Zhang, “Data-intensive applications, challenges, techniques and technologies: A survey on big data,” *Information Sciences*, vol. 275, pp. 314–347, 2014.

[2] R. Clarke, H. W. Ressom, A. Wang, J. Xuan, M. C. Liu, E. A. Gehan, and Y. Wang, “The properties of high-dimensional data spaces: implications for exploring gene and protein expression data,” *Nature Reviews Cancer*, vol. 8, no. 1, pp. 37–49, 2008.

[3] C. Giraud, *Introduction to high-dimensional statistics*. CRC Press, 2014, vol. 138.

[4] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[6] J. Fan, F. Han, and H. Liu, “Challenges of big data analysis,” *National science review*, vol. 1, no. 2, pp. 293–314, 2014.

[7] G. J. Szekely, M. L. Rizzo, N. K. Bakirov *et al.*, “Measuring and testing dependence by correlation of distances,” *The annals of statistics*, vol. 35, no. 6, pp. 2769–2794, 2007.

[8] G. J. Székely and M. L. Rizzo, "The distance correlation t-test of independence in high dimension,’" *Journal of Multivariate Analysis*, vol. 117, pp. 193–213, 2013.

[9] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[10] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, p. 436, 2015.

[11] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2002.

[12] R. A. Johnson and D. W. Wichern, *Applied multivariate statistical analysis*. Prentice hall Englewood Cliffs, NJ, 1992, vol. 4.

[13] I. K. Fodor, “A survey of dimension reduction techniques,” 2002.

[14] M. Balasubramanian and E. L. Schwartz, “The isomap algorithm and topological stability,” *Science*, vol. 295, no. 5552, pp. 7–7, 2002.

[15] S. T. Roweis and L. K. Saul, “Nonlinear dimensionality reduction by locally linear embedding,” *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[16] D. L. Donoho and C. Grimes, “Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data,” *Proceedings of the National Academy of Sciences*, vol. 100, no. 10, pp. 5591–5596, 2003.

[17] M. Belkin and P. Niyogi, “Laplacian eigenmaps for dimensionality reduction and data representation,” *Neural computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[18] H. Feng *et al.*, “Gene classification using parameter-free semi-supervised manifold learning,” *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 9, no. 3, pp. 818–827, 2012.

[19] J. Shawe-Taylor and N. Cristianini, *Kernel methods for pattern analysis*.Cambridge university press, 2004.

[20] S. David, S. Ruey *et al.*, “Independent component analysis via distance covariance,” *Journal of the American Statistical Association*, 2017.

[21] K. P. Adragni, E. Al-Najjar, S. Martin, S. K. Popuri, and A. M. Raim, “Groupwise sufficient dimension reduction with principal fitted components,” *Computational Statistics*, vol. 31, no. 3, pp. 923–941, 2016.

[22] Z. Guo, L. Li, W. Lu, and B. Li, “Groupwise dimension reduction via envelope method,” *Journal of the American Statistical Association*, vol. 110, no. 512, pp. 1515–1527, 2015.

[23] A. D. Ward and G. Hamarneh, “The groupwise medial axis transform for fuzzy skeletonization and pruning,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 32, no. 6, pp. 1084–1096, 2010.

[24] J. Zhou, J. Wu, and L. Zhu, “Overlapped groupwise dimension reduction,” *Science China Mathematics*, vol. 59, no. 12, pp. 2543–2560, 2016.

[25] J. Gui, S.-L. Wang, and Y.-K. Lei, “Multi-step dimensionality reduction and semi-supervised graph-based tumor classification using gene expression data,” *Artificial intelligence in medicine*, vol. 50, no. 3, pp. 181–191, 2010.

[26] M. Brand, “Fast low-rank modifications of the thin singular value decomposition,” *Linear algebra and its applications*, vol. 415, no. 1, pp. 20–30, 2006.

[27] L. v. d. Maaten and G. Hinton, “Visualizing data using t-sne,” *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[28] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[29] D. M. Witten, R. Tibshirani, and T. Hastie, “A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis,” *Biostatistics*, vol. 10, no. 3, pp. 515–534, 2009.

[30] Z. Wu, Y. Li, A. Plaza, J. Li, F. Xiao, and Z. Wei, “Parallel and distributed dimensionality reduction of hyperspectral data on cloud computing architectures,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 9, no. 6, pp. 2270–2278, 2016.

[31] M. Betancourt, “A conceptual introduction to hamiltonian monte carlo,” *arXiv preprint arXiv:1701.02434*, 2017.

[32] A. Soylemezoglu, S. Jagannathan, and C. Saygin, “Mahalanobis taguchi system (mts) as a prognostics tool for rolling element bearing failures,” *Journal of Manufacturing Science and Engineering*, vol. 132, no. 5, p. 051014, 2010.

[33] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available:

[34] Y. Bulatov, “Notmnist dataset,” *Google (Books/OCR), Tech. Rep.[Online].*, 2011.

[35] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror, “Result analysis of the nips 2003 feature selection challenge,” in *Advances in neural information processing systems*, 2005, pp. 545–552.

[36] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, “Imagenet large scale visual recognition challenge,” *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[37] R. Silipo, I. Adae, A. Hart, and M. Berthold, “Seven techniques for dimensionality reduction,” *KNIME*, 2014.

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, “Scikit-learn: Machine learning in python,” *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[39] A. Frieze, R. Kannan, and S. Vempala, “Fast monte-carlo algorithms for finding low-rank approximations,” *Journal of the ACM (JACM)*, vol. 51, no. 6, pp. 1025–1041, 2004.

[40] X. Huo and G. J. Székely, “Fast computing for distance covariance,” *Technometrics*, vol. 58, no. 4, pp. 435–447, 2016. [Online].

# III. DIRECT ERROR DRIVEN LEARNING FOR DEEP NEURAL NETWORKS WITH APPLICATIONS TO BIG DATA

R. Krishnan, S. Jagannathan

Department of Electrical & Computer Engineering

Missouri University of Science and Technology

Rolla, Missouri 65409–0050

Email: krm9c, sarangap@mst.edu

V.A. Samaranayake

Department of Mathematics & Statistics

Missouri University of Science and Technology

Rolla, Missouri 65409–0050

Email: vsam@mst.edu

## ABSTRACT

In this paper, heterogeneity and noise in big-data are shown to increase the generalization error for a traditional learning regime utilized for deep neural networks (deep NN). To reduce this error while overcoming the issue of vanishing gradients, a direct error-driven learning scheme is proposed. First, to reduce the impact of heterogeneity and data-noise, the concept of a neighborhood is introduced. Using this neighborhood, an approximation of generalization error is obtained and an overall error, comprised of learning and the approximate generalization errors, is defined. A novel NN weight-tuning law is obtained through a layer-wise performance measure enabling the direct use of overall error for learning. Additional constraints are introduced into the layer-wise performance measure to guide and improve the learning process in the presence of noisy dimensions. The proposed direct error-driven learning scheme effectively addresses the issue of heterogeneity and noise while mitigating vanishing gradients and noisy dimensions. A comprehensive simulation study is presented where the proposed approach is shown to mitigate the vanishing gradient problem while improving generalization by 6%.

## 1. INTRODUCTION

Classification in the presence of large data-sets is a common problem in many fields today where many challenges must be addressed for efficiency [1]. These challenges include (1) heterogeneity of two types: semantic heterogeneity: where unique characteristics arise due to the diversity among data-sources, and statistical heterogeneity- dissimilarity between statistical properties of any one part of a dataset to other parts of the data-set [1]; (2) data-noise- where data-points may be distorted; (3) noisy dimensions: those not contributing to the actual prediction [2, 1] and (4) spurious correlations- unwanted relationships in the data.

Deep neural networks (deep NN) can address a few of these challenges, where a composition of linear and nonlinear transformations [3] with tunable parameters is present. Typically, these parameters are estimated using a performance measure that is minimized through stochastic gradient descent (SGD) [4]. In spite of many successes, noise and heterogeneity [2, 1, 5] are challenges and can increase generalization error in deepNN-based approaches.

In the literature, cross-validation methods [6], norm regularization [7] and dropouts [8] are commonly used to compensate for these effects. However, these methods [7, 8] depend on the practitioner's experience and cannot guarantee performance. Furthermore, determining the hyper-parameters for the classifier depends on upper-bounds for generalization error. Generally, these upper-bounds are derived using Vapnik-Chervonenki's dimensions, Radmacher complexity or uniform stability [4, 9]. However, this process often requires complex mathematical analysis and the bounds are model dependent. In brief, the approaches present in the literature [6, 9, 8, 10] are either heuristic and do not guarantee low generalization error or require an exact understanding of the underlying data-distribution for efficiency in the presence of noise and heterogeneity.

On the other hand, SGD [11] learning suffers from unstable learning due to diminishing or exploding learning signals in deep neural networks [12, 13]. Scaling the gradients using relu activation function [14, 12], learning rate schedules [15] and $L_1/L_2$ regularization [16] methods with robust weight initializations [13] are common ways of addressing this issue. However, no guarantee can be given regarding diminishing learning signals and the use of different activation functions may not solve the problem completely[14].

To overcome the vanishing/exploding gradient, an alternate learning scheme, known as target propagation, was introduced in [17]. In [17], auto-encoders are used to estimate targets at every layer for learning. However, it is difficult in most cases to define targets at each layer and the computational overload of maintaining auto-encoders at every layer is large. To alleviate this issue, direct feedback alignment was proposed in [18] where the error is directly used for learning at every layer. Although impressive results have been reported in [18], there are some drawbacks of the approach. First, the approach projects the error in random directions which allows the learning to progress at random. Second, learning would be inhibited in the presence of a zero-derivative activation function.

Motivated by these challenges, this paper proposes a systematic approach for addressing heterogeneity and noise in big-data with deep NN-based classification as an application. Specifically, the impact of big data challenges are addressed by (1) reducing the generalization error due to heterogeneity and data-noise, (2) improving the learning efficiency in the presence of noisy dimensions, and (3) mitigating vanishing gradients.

For every sample of data, the cost of learning is first estimated by feeding the data-sample into the classifier. Next, synthetic distortions are introduced to create a neighborhood around the sample which is utilized to approximate the cost due to the generalization error. Next, an overall cost is defined, comprised of learning and generalization errors. A learning problem is then designed to minimized the overall cost. By minimizing generalization error in the learning procedure, the impact of heterogeneity and data-noise is explicitly mitigated in contrast with [19, 20, 21, 8, 10]. Moreover, in contrast with [9], the underlying distribution of data need not be known to calculate the approximated generalization cost.

To learn the classifier, a direct error-driven learning (EDL) regime is proposed, where the weights at each layer in the deep NN learn directly from the overall error, which is comprised of learning and generalization errors. To enable this learning, a cost function is defined at each layer in the deep NN as a function of the overall cost. In contrast with traditional gradient descent [4, 7, 12], the feedback is designed to ensure that the magnitude of the learning signal does not vanish. Finally, a novel weight-update rule is obtained using the derivative of the layer-wise cost. Therefore, the update rule is generic enough to solve any cost function and can address the issue of heterogeneity and data-noise through the cost function. The direct use of error for learning also mitigates the issue of vanishing gradients.

The deep NN weight-update law also includes a decay term to counter the issue of ineffective learning in the presence of noisy dimensions. To obtain the decay term, singular values of the correlation matrix are utilized to determine the dimensions that are noisy. Finally, penalties are introduced into the learning rule to inhibit learning in the directions influenced by these dimensions. In traditional regularization methods [7, 8, 10], a constant penalty is injected in all directions, therefore, one cannot effect learning in specific directions. In contrast, the penalties in the EDL are proportional to the measure of the variance provided by the singular values which discourage learning in noisy directions while encouraging it along others.

The direct error-driven learning regime may look similar to tracking error-based NN updates [22] or direct feedback alignment [18] due to the use of a user-defined matrix for projecting the error . However, there are some key improvements: (1) the user-defined matrix is specifically designed to propel learning in the directions of steepest descent and (2) the learning is not inhibited even when the activation function is saturated. A comprehensive simulation study using ten benchmarking data-sets is finally presented for testing the proposed approach. Simulation indicates that the proposed methodology mitigates the problem of vanishing gradients and outperforms SGD in the presence of heterogeneity and noise.

The contributions of this paper therefore include: (1) the development of a novel cost formulation for optimization of deep NN-based classification to reduce the impact of heterogeneity and data-noise; (2) Using the cost function, a direct EDL scheme is defined to minimize the cost function to address heterogeneity and data-noise while mitigating the vanishing gradient problem; (3) the design of a systematic approach to mitigate the impact of noisy dimensions to learning for the EDL scheme and (4) the verification of the efficacy of the proposed approach with simulation analysis.

The rest of the paper is organized as follows. Section II describes the problem statement and establishes the motivations for the paper. Section III describes the proposed framework and the direct error-driven learning scheme while Section IV outlines simulation results for the paper. Finally, Section V outlines the conclusions.

## 2. PROBLEM STATEMENT

In this section, the notations used in this paper are introduced first. Next, the objectives of classification are presented and finally, the impact of big-data challenges on generalization error are demonstrated. Throughout the paper, matrices and vectors are denoted by boldface type; lowercase symbols are used for vectors; and uppercase symbols are used for matrices. Furthermore, superscripts $(i)$ indicate the layer index for the deep NN. The iteration index in the learning phase and the prediction phase are denoted by $t$ and $k$ respectively. Squared Frobenius norm for matrices and euclidean norm for vectors are denoted by $||.||$.

Consider any data generating process, e.g. manufacturing system, where $\boldsymbol{x}_t \in \mathbb{R}^{p\times n}$ represents the data, at the sampling instant $t$. Let the sample $\boldsymbol{x}_t$ be collected from $p$ attributes with $n$ being the number of sample points. The objective is to determine whether the system is in the healthy state or at one of the $\mathcal{F}$ faults using $\boldsymbol{x}_t$ via an input-output map. Assuming that $\phi(.)$ can be approximated using a deep neural network (deep NN) with the weights that includes layer-wise bias denoted as $\boldsymbol{\theta} = [\boldsymbol{W}^{(1)} \cdots \boldsymbol{W}^{(d)}]$, it follows from the approximation property of NNs [23] that

$$\begin{aligned} \phi(\boldsymbol{x}_t) = & \quad \boldsymbol{y}(\boldsymbol{x}_t;\boldsymbol{\theta}) + \varepsilon, \\ = & \quad f^{(d)}(\boldsymbol{W}^{(d)} \cdots (f^{(1)}(\boldsymbol{W}^{(1)}(\boldsymbol{x}_t))))) + \varepsilon, \end{aligned}$$

where $\boldsymbol{y} \in \mathbb{R}^{\mathcal{F}\times n}$ represents the ideal output for $\phi(.)$. The term $\varepsilon$ represents the approximation error and $f^{(i)}, \forall \quad i = 1 \cdots d$ denote the layer-wise activation functions with $d$ layers. Let the estimated map $\hat{\boldsymbol{y}}(\boldsymbol{x}_t)$ with weights denoted as $\hat{\boldsymbol{\theta}} = [\hat{\boldsymbol{W}}^{(1)} \cdots \hat{\boldsymbol{W}}^{(d)}]$ be written as

$$\hat{\boldsymbol{y}}(\boldsymbol{x}_t) = \quad \hat{\boldsymbol{y}}(\boldsymbol{x}_t;\hat{\boldsymbol{\theta}}), = f^{(d)}(\hat{\boldsymbol{W}}^{(d)} \cdots (f^{(1)}(\hat{\boldsymbol{W}}^{(1)}\boldsymbol{x}_t)))). \tag{1}$$

To learn the weights, a data-set $\mathcal{D}$ is defined by collecting a representative sample from each of the fault populations denoted as $\psi_i, \forall i = 1, 2, \cdots \mathcal{F}$ where the following assumptions hold.

**Assumption 6** *Samples from each fault are obtained such that they are independently and identically distributed.*

**Assumption 7** *The distribution of each of the categories in the learning phase is similar to the distribution of the data in the prediction phase.*

Each data-point in $\mathcal{D}$ is collected across $p$ attributes and labels are vectors of $\mathbb{R}^{\mathcal{F}\times n}$. Learning $\phi(.)$ via $\hat{\boldsymbol{y}}(\boldsymbol{x}_t;\hat{\boldsymbol{\theta}})$ involves minimizing a cost function $J_o(.)$ while estimating $\hat{\boldsymbol{\theta}}$ [24]. Let the expected value of the learning error $\boldsymbol{e}_l$ be defined as

$$E[\boldsymbol{e}_l(k)] = E_{\forall \boldsymbol{x}\in\mathcal{X}, \boldsymbol{y}\in\mathcal{Y}}[\boldsymbol{y}(\boldsymbol{x};\boldsymbol{\theta}) - \hat{\boldsymbol{y}}(\boldsymbol{x};\hat{\boldsymbol{\theta}})],$$

where $E[.]$ refers to the expectation operator. For simplicity, the iteration index $k$ is only indicated when weight update laws are

$$E[\boldsymbol{e}] = \quad E[\boldsymbol{e}_l] + \varepsilon, \tag{2}$$

where $E[\boldsymbol{e}]$ is the total expected error and $\varepsilon$ is the approximation error.

**Remark 17** *Note that $\varepsilon = \phi(x(t)) - \boldsymbol{y}(\boldsymbol{x}_t;\boldsymbol{\theta})$, is understood as the error between NN approximation $\boldsymbol{y}(\boldsymbol{x}_t;\boldsymbol{\theta})$ with ideal parameters $\boldsymbol{\theta}$ and the underlying function $\phi$. Therefore, $\varepsilon$ depends on the choice of NN parameters (the number of hidden layer neurons and activation function) and the properties of the underlying function $\phi$ (differentiability, complexity class, etc. [25]). Resultantly, $\varepsilon$ is assumed as the approximation error corresponding to the NN model that is considered for analysis. Therefore, $\varepsilon$ is assumed constant such that $E[\varepsilon] = \varepsilon$ in Eq.* (2).

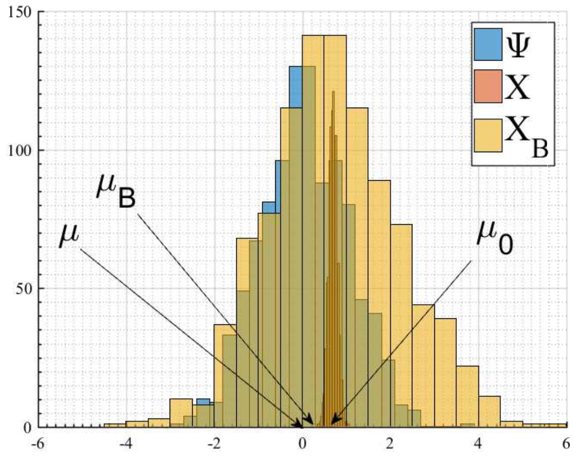
The overall cost $J(\hat{\boldsymbol{\theta}})$ is obtained as

$$\begin{aligned} J(\hat{\boldsymbol{\theta}}) = E[\boldsymbol{e}]^T E[\boldsymbol{e}], = &\quad E[\boldsymbol{e}_l]^T E[\boldsymbol{e}_l] + 2\varepsilon^T E[\boldsymbol{e}_l] + \varepsilon^T \varepsilon, \\ = &\quad J_{emp} + J_{gen} + J_{apx}, \end{aligned}$$

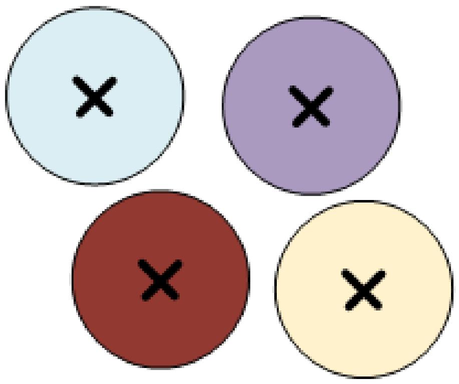
where $J_{emp} = E[\boldsymbol{e}_l]^T E[\boldsymbol{e}_l]$ is the empirical cost with $J_{gen} = 2\varepsilon^T E[\boldsymbol{e}_l]$ and $J_{apx} = \varepsilon^T \varepsilon$ being the cost introduced by generalization and approximation errors, respectively [25]. Usually, the unknown quantities $J_{gen}$ and $J_{apx}$ are assumed to be bounded [25] while $J_{emp}$ is minimized during learning.

However, $J_{gen}$ increases due to heterogeneity and noise [1] and cannot be assumed to be bounded. For illustration, let big-data be collected from one attribute and be stored at multiple locations. Let the healthy case be denoted by $\Psi$ with mean $\mu$ as shown in Figure 1b and consider $\boldsymbol{\mathcal{X}}$ with mean $\mu_0$ to be sampled from one of the available data-storage locations. Because, there are certain characteristics of $\Psi$ not covered by $\boldsymbol{\mathcal{X}}$, it does not represent $\Psi$ fully as seen in Figure 1a. As a result, methodologies that learn with $\boldsymbol{\mathcal{X}}$ do not capture $\Psi$ fully, which results in erroneous predictions and is reflected in an increased $J_{gen}$.

On the other hand, in the presence of noisy dimensions, learning is focused in directions that may not be useful [26]. This is observed in Figure 3, when the number of noisy dimensions in the data increases, the accuracy falls. A methodology to mitigate the aforementioned challenges is needed and is presented next.



(a)



(b)

Figure 1. (a) Distribution of the actual, the first sample and the second sample that is $\psi, \boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{X}}_B$ (b) Illustration for the neighborhood principle.

## 3. ERROR-DRIVEN LEARNING

In this approach, synthetic distortions are used to define a neighborhood around the data-point which is utilized to generate an approximation of generalization error. An optimization approach is designed to minimize the learning and generalization errors using a novel update law. Subsequently, the impact of noisy dimensions is addressed with an additional term introduced into the update law. The details of this proposed methodology are provided in the following sub-sections. The impact of heterogeneity and data-noise is discussed first.

## 3.1. MITIGATING HETEROGENEITY AND NOISE

As mentioned earlier, the challenge of heterogeneity and data-noise arises because the data used during the learning phase does not present the complete picture of the underlying problem. Thus, in both these cases, the classifier may encounter data during the prediction phase with characteristics that were not described by the training sample. These data-points are referred to as "unseen samples," and they result in an increased $J_{gen}$.

To approximate $J_{gen}$, one needs to approximate the unseen samples. Therefore, let every data-point in the training set be a center of a neighborhood such that the data-points in the neighborhood may be understood as unseen samples that are similar to the center point. An example is illustrated in Figure 1b, where the "$\times$" denotes the center and the circle denotes the neighborhood. Under the assumption that these unseen samples belong to the same category as the center, generalization error $\boldsymbol{e}_{gen}$ and its associated cost $J_{gen}$ can be approximated using the neighborhood.

Formally, let $\Delta \boldsymbol{x}_t \sim p'(\boldsymbol{\mu}, \boldsymbol{\mathcal{S}})$ represent perturbations introduced into every data-point $\boldsymbol{x}$, where $p'(.)$ is a probability distribution of choice characterized by the first two moments: mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\mathcal{S}}$. Thus, for every $\boldsymbol{x} \in \boldsymbol{\mathcal{X}}$, there exists a $\boldsymbol{x} + \Delta \boldsymbol{x}$ with $||\Delta \boldsymbol{x}|| \leq Q$ such that $\boldsymbol{x} + \Delta \boldsymbol{x}$ belongs to the same category as $\boldsymbol{x}$. A collection of $\boldsymbol{x} + \Delta \boldsymbol{x}$ represents the neighborhood for $\boldsymbol{x}$. The collection of the data-points in the neighborhood for all $\boldsymbol{x}$ is written as

$$\boldsymbol{\mathcal{X}}_B = \{\boldsymbol{x}_B | \boldsymbol{x}_B = \boldsymbol{x} + \Delta \boldsymbol{x}, \forall 0 \leq ||\Delta \boldsymbol{x}|| \leq Q, \forall \boldsymbol{x} \in \boldsymbol{\mathcal{X}}\}.$$

For an example, the histogram of $\mathcal{X}_B$ illustrated in Figure 1b is the collection of all the data-samples obtained when Gaussian noise with zero mean and unit variance is introduced into $\mathcal{X}$. It can be observed that $\mathcal{X}_B$ is a better representative of $\Psi$ than $\mathcal{X}$.

Depending on the magnitude of the perturbations, the neighborhood could represent the entire input space or a small region around $\boldsymbol{x}$. However, if the magnitude of $\Delta\boldsymbol{x}$ is large, $\mathcal{X}_B$ might also include outliers in the distribution. On the other hand, if the magnitude of the perturbations is small, large generalization error may be observed due to the overlapping of neighborhoods as is observed in Figure 1b. The trade-off can be addressed by carefully choosing the size of the neighborhood. More details and simulation results on the significance of the neighborhood are presented in the appendix.

Ideally, it is desirable to make the classifier resilient to the presence of all the unseen samples. Thus, the cost of generalization is incorporated into the learning problem by defining $E[\boldsymbol{e}_{gen}]^T E[\boldsymbol{e}_{gen}]$ as $\hat{J}_{gen}$ with

$$E[\boldsymbol{e}_{gen}] = E_{\forall \boldsymbol{x}_B \in \mathcal{X}_B, \boldsymbol{y} \in \boldsymbol{\mathcal{Y}}}[\boldsymbol{y}(\boldsymbol{x};\boldsymbol{\theta}) - \hat{\boldsymbol{y}}(\boldsymbol{x}_B;\hat{\boldsymbol{\theta}})].$$

Substitution into Eq. (3) reveals

$$\begin{aligned} J(\hat{\boldsymbol{\theta}}) &= \left(J_{emp} + \hat{J}_{gen}\right) + J_{apx}, \\ &= \left(E[\boldsymbol{e}_l]^T E[\boldsymbol{e}_l] + E[\boldsymbol{e}_{gen}]^T E[\boldsymbol{e}_{gen}]\right) + J_{apx}, \end{aligned}$$

with $\boldsymbol{e}_l$ being the estimated learning error and $\boldsymbol{e}_{gen}$ being the generalization error. The factor $J_{apx}$ is assumed to be bounded and constant. For brevity of notation, the expectation operators are not explicitly stated from hereon.

The learning problem thus involves the minimization of $J$ with respect to the estimated weights $\hat{\boldsymbol{\theta}}$ such that

$$\boldsymbol{\theta}^* = \underset{\hat{\boldsymbol{\theta}} \in \boldsymbol{\Omega}}{\operatorname{argmin}} J(\hat{\boldsymbol{\theta}}), \qquad (3)$$

with $\boldsymbol{\Omega}$ representing the parameter space. Introducing perturbations allows the methodology to capture more of the data-distribution than what is represented by $\boldsymbol{\mathcal{X}}$, as observed from the histogram of $\boldsymbol{\mathcal{X}}_B$ in Figure 1b. Thus, $\boldsymbol{\mathcal{X}}_B$ approximates the impact of heterogeneity rather than knowing it a priori, in contrast with [27]. In contrast with cross-validation [9] and regularization approaches [16], the proposed approach introduces model selection into the optimization problem such that the learning procedure can methodically establish a trade-off between the learning error, bias of the model towards the training data, and the generalization error which is the variance of the model. To optimize Eq. (3), a learning approach is presented next.

## 3.2. LEARNING APPROACH

To optimize Eq. (3), the following update law is defined as

$$\hat{\boldsymbol{W}}_{k+1}^{(i)} = \hat{\boldsymbol{W}}_{k}^{(i)} - \alpha \boldsymbol{u}_{k}^{(i)}, \tag{4}$$

where $\alpha > 0$ is the learning rate. An $L_2$ norm regularized cost function is defined to ensure the boundedness of the weights where

$$H(\hat{\boldsymbol{y}}, \boldsymbol{x}; \hat{\boldsymbol{\theta}}) = \left[ J(\hat{\boldsymbol{\theta}}) + \lambda \sum_{i=1}^{d} \|\boldsymbol{W}_{k}^{(i)}\| \right], \tag{5}$$

with $\lambda > 0$ being the decay coefficient. The learning problem is finally given as re $\alpha > 0$ is the learning rate. An $L_2$ norm regularized cost function is defined to ensure the boundedness of the weights where

$$\boldsymbol{\theta}^* = \underset{\hat{\theta} \in \boldsymbol{\Omega}}{\arg\min} \quad H(\hat{\boldsymbol{y}}, \boldsymbol{x}; \hat{\boldsymbol{\theta}}). \tag{6}$$

Note that minimizing Eq. (6) is equivalent to minimizing Eq. (3) with the weight constraints. The weight updates at each iteration $k$ are given as

$$\boldsymbol{u}_k^{(i)} = \nabla_{\hat{\boldsymbol{W}}_k^{(i)}} H(\hat{\boldsymbol{y}}, \boldsymbol{x}; \theta), = \boldsymbol{\delta}_k^{(i)} + \lambda \hat{\boldsymbol{W}}_k^{(i)}. \tag{7}$$

Let the term $\nabla_{\hat{\boldsymbol{W}}_k^{(i)}}(.)$ denote the gradient of $(.)$ with respect to the NN weight $\hat{\boldsymbol{W}}_k^{(i)}$ which results in $\boldsymbol{\delta}_k^{(i)}$ as

$$\boldsymbol{\delta}_k^{(i)} = \nabla_{\hat{\boldsymbol{W}}_t^{(i)}} \boldsymbol{J} = \boldsymbol{G}^{(i)}(\boldsymbol{x}) + \boldsymbol{G}^{(i)}(\boldsymbol{x} + \Delta \boldsymbol{x}), \tag{8}$$

where $G^{(i)}(\boldsymbol{x}) = \nabla_{\hat{\boldsymbol{W}}_t^{(i)}} J_{emp}$ represents the gradient of the empirical cost and $G^{(i)}(\boldsymbol{x} + \Delta x) = \nabla_{\hat{\boldsymbol{W}}_t^{(i)}} \hat{J}_{gen}$ represents the gradient of the cost due to generalization error. Applying the chain rule, a generalized expression for $\boldsymbol{G}^{(i)}(\boldsymbol{x})$ can be derived as

$$\begin{gathered} \boldsymbol{G}^{(i)}(\boldsymbol{x}) = -f^{(i-1)}(\boldsymbol{x}) \boldsymbol{e}_l^T \\ \left[ \prod_{j=d}^{i+1} diag(\nabla f^{(j)}(\boldsymbol{x})) \hat{\boldsymbol{W}}^{(j)} \right] diag(\nabla f^{(i)}(\boldsymbol{x})). \end{gathered} \tag{9}$$

Denote $\prod_{j=d}^{i+1}(diag(\nabla f^{(j)}(\boldsymbol{x}))\hat{\boldsymbol{W}}^{(j)}) diag(\nabla f^{(i)}(\boldsymbol{x}))$ as $\boldsymbol{\mathcal{T}}^{(i)}(\boldsymbol{x})$ and simplify Eq. (8) to get

$$\boldsymbol{\delta}_k^{(i)} = -[\underbrace{f^{(i-1)}(\boldsymbol{x})}_{\eta^{(i-1)} \times 1} \quad \underbrace{f^{(i-1)}(\boldsymbol{x} + \Delta \boldsymbol{x})}_{\eta^{(i-1)} \times 1}] \boldsymbol{\epsilon}^T \boldsymbol{\mathcal{T}}^{(i)}, \tag{10}$$

where the overall error is written as

$$\boldsymbol{\epsilon}^T = diag(\frac{dJ(\boldsymbol{\theta})}{d\hat{\boldsymbol{y}}})^T = diag(\underbrace{\boldsymbol{e}_l}_{\mathcal{F} \times 1}, \underbrace{\boldsymbol{e}_{gen}}_{\mathcal{F} \times 1}),$$

where $J(\boldsymbol{\theta})$ is defined in Eq. (3) and the expectation operators are ignored. Similarly, by denoting $\prod_{j=d}^{i+1}(diag(\nabla f^{(j)}(\boldsymbol{x} + \Delta \boldsymbol{x}))\hat{\boldsymbol{W}}^{(j)}) diag(\nabla f^{(i)}(\boldsymbol{x} + \Delta \boldsymbol{x}))$ as $\boldsymbol{\mathcal{T}}^{(i)}(\boldsymbol{x} + \Delta \boldsymbol{x})$ we get

$$\boldsymbol{\mathcal{T}}^{(i)} = diag(\underbrace{\boldsymbol{\mathcal{T}}^{(i)}(\boldsymbol{x})}_{\mathcal{F} \times \eta^{(i)}}, \underbrace{\boldsymbol{\mathcal{T}}^{(i)}(\boldsymbol{x} + \Delta \boldsymbol{x})}_{\mathcal{F} \times \eta^{(i)}}),$$

where $\eta^{(i)}$ represents the number of neurons in the hidden layer $(i)$.

Observe that the overall error $\boldsymbol{\epsilon}$ has to propagate through transformation $\boldsymbol{\mathcal{T}}^{(i)}$ to impact learning. Since $\boldsymbol{\mathcal{T}}^{(i)}(\boldsymbol{x})$ and $\boldsymbol{\mathcal{T}}^{(i)}(\boldsymbol{x} + \Delta\boldsymbol{x}))$ are linear approximations in the neighborhood of the weights, the singular vectors of $\boldsymbol{\mathcal{T}}^{(i)}$ dictate the directions of learning and their magnitude is described by the singular values of $\boldsymbol{\mathcal{T}}^{(i)}$. Each diagonal element in $\boldsymbol{\mathcal{T}}^{(i)}$ is the product of the derivative of the layer-wise activation functions. In other words, the singular values of $\boldsymbol{\mathcal{T}}^{(i)}$ would also tend to zero as the number of layers in the deep NN increase, otherwise known as the vanishing gradients issue [12].

In the literature, ResNet [28] and DenseNet [29] have been demonstrated to present improved performance in the presence of vanishing gradients. In these approaches[28, 29] , changes are introduced into the NN architecture such that the gradients do not vanish. However, specific architectures are employed for analysis and these approaches are not generic. In contrast, the learning approach presented in this paper is not dependent upon the type of architecture used.

Therefore, we propose to directly utilize the error for learning the weights at every layer instead of propagating the error through the hidden layers using $\boldsymbol{\mathcal{T}}^{(i)}$ [4] or random matrices [30]. To enable this learning, a layer-wise performance measure $H^{(i)}(\hat{\boldsymbol{y}}, \boldsymbol{x}; \hat{\boldsymbol{W}}^{(i)})$ is defined and $H^{(i)}(\hat{\boldsymbol{y}}, \boldsymbol{x}; \hat{\boldsymbol{W}}^{(i)})$ is optimized for each layer instead of the overall cost. Specifically, the layer-wise cost is defined as a function of overall error through $\boldsymbol{\delta}_k^{(i)}$ such that minimizing $H^{(i)}(\hat{\boldsymbol{y}}, \boldsymbol{x}; \hat{\boldsymbol{W}}^{(i)})$ for each $i$ minimizes the overall error. It follows that

$$H^{(i)}(\hat{\boldsymbol{W}}^{(i)}) = \frac{1}{2}\left[tr((\boldsymbol{\delta}_k^{(i)})^T \boldsymbol{P} \hat{\boldsymbol{W}}_k^{(i)}) + \lambda \|\boldsymbol{W}_k^{(i)}\|^2\right], \tag{11}$$

where $tr(.)$ is the trace operator and $\boldsymbol{P}$ is a positive definite symmetric matrix of choice. Note that for each layer $i$, $\boldsymbol{\delta}_k^{(i)}$ can be understood as the feedback provided by the overall cost $J(\boldsymbol{\theta})$ towards controlling the layer $(i)$. Therefore, the layer-wise cost can be interpreted as the minimization of the correlation between $\boldsymbol{\delta}_k^{(i)}$ and $\hat{\boldsymbol{W}}_k^{(i)}$ under the constraint that $\|\hat{\boldsymbol{W}}_k^{(i)}\|$ must be as small as possible.

**Remark 18** *Observe that*

$$tr((\boldsymbol{\delta}_k^{(i)})^T \boldsymbol{P}\hat{\boldsymbol{W}}_k^{(i)}) = \quad tr((\boldsymbol{P}\hat{\boldsymbol{W}}_k^{(i)}(\boldsymbol{\delta}_k^{(i)})^T) = tr(\boldsymbol{P}\boldsymbol{X})$$

*where* $\boldsymbol{X} = \hat{\boldsymbol{W}}_k^{(i)}(\boldsymbol{\delta}_k^{(i)})^T$ *is a square matrix and* $\boldsymbol{P}$ *is symmetric and positive definite matrix. Rewrite* $\boldsymbol{P} = \sum_{i=1}^{\eta^{(i)}} \nu_i \boldsymbol{p}_i^T \boldsymbol{p}_i$. *Substitution into the equation above and simplification reveals* $tr(\boldsymbol{P}\boldsymbol{X}) = \sum_{i=1}^{\eta^{(i)}} \nu_i(tr(\boldsymbol{p}_i \boldsymbol{X} \boldsymbol{p}_i^T)))$, *which is a linear combination of non-negative values [31]. Thus, the formulation in Eq.* (11) *can be alternatively interpreted as a linear combination of standard quadratic forms that are independently convex [31].*

To define $\boldsymbol{\delta}_k^{(i)}$ at each layer, $\boldsymbol{\mathcal{T}}^{(i)}$ in Eq. (10) is replaced by a user-defined feedback matrix denoted as

$$\boldsymbol{B}^{(i)} = diag(\underbrace{\boldsymbol{B}^{(i)}(\boldsymbol{x})}_{\mathcal{F}\times\eta^{(i)}}, \underbrace{\boldsymbol{B}^{(i)}(\boldsymbol{x}+\Delta\boldsymbol{x})}_{\mathcal{F}\times\eta^{(i)}}),$$

where $\boldsymbol{B}^{(i)}$ is chosen such that $\boldsymbol{B}^{(i)}(\boldsymbol{B}^{(i)})^T$ is positive definite. The feedback $\boldsymbol{\delta}_k^{(i)}$ is then given as

$$\boldsymbol{\delta}_k^{(i)} = [\underbrace{f^{(i-1)}(\boldsymbol{x})}_{\eta^{(i-1)}\times 1} \quad \underbrace{f^{(i-1)}(\boldsymbol{x}+\Delta\boldsymbol{x})}_{\eta^{(i-1)}\times 1}]\boldsymbol{\epsilon}^T \boldsymbol{B}^{(i)}. \tag{12}$$

Finally, the overall cost may be written as the sum of layer-wise costs

$$H(\hat{\boldsymbol{y}}, \boldsymbol{x}; \hat{\boldsymbol{\theta}}) = \sum_{i=1}^{d} H^{(i)}(\hat{\boldsymbol{W}}^{(i)}). \tag{13}$$

With the cost defined in Eq. (11), the weight updates are defined as $\boldsymbol{u}_k^{(i)} = \nabla_{\hat{\boldsymbol{W}}_k^{(i)}} H(\hat{\boldsymbol{y}}, \boldsymbol{x}; \hat{\boldsymbol{\theta}})$ which results in the update rule as defined in Eq, (7) with $\boldsymbol{\delta}_k^{(i)}$ as defined in Eq. (12).

Observe that the update rule demonstrated in Eq. (12) can be incorporated for any cost function that can be setup for a deep neural network. Thus, the proposed learning regime is composed of a generic and simple weight update rule that can enable learning in most deep NN environments.

The overall $H(\hat{\boldsymbol{y}}, \boldsymbol{x}; \hat{\boldsymbol{\theta}})$ is minimized if each of the layer-wise cost functions are minimized. The stationary condition for each layer-wise cost depends on the term $\boldsymbol{\delta}_k^{(i)}$ since $\frac{d\left(tr\left((\boldsymbol{\delta}_k^{(i)})^T \boldsymbol{P} \hat{\boldsymbol{W}}_k^{(i)}\right)\right)}{d\hat{\boldsymbol{W}}_k^{(i)}} = ((\boldsymbol{\delta}_k^{(i)})^T \boldsymbol{P})^T = \boldsymbol{P}\boldsymbol{\delta}_k^{(i)}$. The term $\boldsymbol{\delta}_k^{(i)}$ is a linear transformation of the overall error $\boldsymbol{\epsilon}$. Observe that $\boldsymbol{\delta}_k^{(i)}$ is zero when the overall error is zero, provided $\boldsymbol{B}^{(i)}$ and $[f^{(i-1)}(\boldsymbol{x}) \quad f^{(i-1)}(\boldsymbol{x} + \Delta\boldsymbol{x})]$ are assumed to be non-zero. When these conditions are satisfied, learning in the direction of minimizing the layer-wise cost function would result in minimizing the overall error. Since, the transformation parameters $\boldsymbol{B}^{(i)}$ are set by the practitioner and $[f^{(i-1)}(\boldsymbol{x}) \quad f^{(i-1)}(\boldsymbol{x} + \Delta\boldsymbol{x})]$ depends on the data, the non-zero assumption is reasonable and the procedure works well in practice as is observed in the simulation.

*Choice of $\boldsymbol{B}^{(i)}$:* The simplest way to choose $\boldsymbol{B}^{(i)}$ is to sample at random from a pre-selected distribution. By ensuring that $\boldsymbol{B}^{(i)}$ is chosen with all positive singular values, the issue of vanishing gradients will be avoided. The associated drawback is that the behavior of the cost function with respect to changing weights has no impact on learning because learning directly depends on the instantaneous error for every sample and the choice of $\boldsymbol{B}^{(i)}$, which leads to randomized direction of learning. In other words, the learning, mimics a random search propelled by the overall error.

It is therefore desirable to choose $\boldsymbol{B}^{(i)}$ to propel learning in directions of the steepest descent for $J(\hat{\boldsymbol{\theta}})$. One way of choosing $\boldsymbol{B}^{(i)}$ is to use the directions of learning dictated by the singular vectors of $\boldsymbol{\mathcal{T}}^{(i)}$. Moreover, positive singular values are chosen at random to signify the magnitude of learning in every direction. The size of the output vector in the deep NN is $\mathcal{F}$ and the number of neurons in any hidden layer is given by $\eta^{(i)}$. It follows that $\boldsymbol{\mathcal{T}}^{(i)} \in \mathbb{R}^{\mathcal{F}+\eta^{(i)} \times \mathcal{F}+\eta^{(i)}}$, yielding $\boldsymbol{B}^{(i)} = \sum_{j=1}^{\mathcal{F}+\eta^{(i)}} \kappa_j^{(i)} (\boldsymbol{v}_j^{(i)})^T \boldsymbol{v}_j^{(i)}$, where $\boldsymbol{v}_j$ represents the singular-vectors of $\mathcal{T}^{(i)}$ for each $j$. Similarly, $\kappa_j > 0$ is the singular value that is chosen at random for any layer $i$. When $\boldsymbol{B}^{(i)}$ is chosen as described above, $\boldsymbol{B}^{(i)}$ would project the error in the direction of the steepest descent for every layer.

Furthermore as different layers in the NN align themselves towards the steepest descent of the cost function, there is no incubation period for improvement in accuracy and the alignment problem is mitigated, in contrast with [18]. Next, the issue of noisy dimensions is discussed.

## 3.3. MITIGATING NOISY DIMENSIONS

In the presence of noisy dimensions, the hidden layers in the deep NN may generate features that are non-informative and would lead to deterioration in performance [26]. For illustration, assume that the neurons with small output variances generate features that may not be useful at any hidden layer $i$. One may observe that the output at any neuron at layer $i$ is the consequence of corresponding weight connections to that neuron at layer $i$. If the neuron is generating features that are non-informative, learning in the weight connections corresponding to that neuron are less important relative to others. A method to control learning in specific directions is therefore introduced.

Consider, for illustration, a neuron $j$ at hidden layer $i$ and assume that the $j^{th}$ neuron is irrelevant to the final prediction. All the weight connections corresponding to the $j^{th}$ neuron form the $j^{th}$ row in $\hat{\boldsymbol{W}}^{(i)}$. To scale the weight connections corresponding to the $j^{th}$ neuron, pre-multiply the weight matrix by an identity matrix $\boldsymbol{D}^{(i)}$ with a scaling factor on the $j^{th}$ row and $j^{th}$ column. The factor now affects all the elements at row $j$. To introduce a scaling proportional to the amount of variance in that particular neuron, singular values can be used. Subsequently subtract the scaled matrix from the original $\hat{\boldsymbol{W}}^{(i)}$ to introduce decay in specific directions.

To generalize the principles described above, let the output at every layer be represented by $\boldsymbol{X}^{(i)} \in \mathbb{R}^{b \times \eta^{(i)}}$, where $b$ is the size of the sample at each training iteration. Write the singular value decomposition as $\boldsymbol{U}^{(i)}\boldsymbol{D}^{(i)}\boldsymbol{V}^{(i)T}$, where $\boldsymbol{U}^{(i)}$ and $\boldsymbol{V}^{(i)T}$ provide the singular vectors. The term $\boldsymbol{D}^{(i)}$ is the diagonal matrix of all singular values providing a measure of variance for every neuron. Since the scaling factor must be between zero and one, let $\boldsymbol{D}^{(i)}$ be normalized such that the elements sum up to one such that

$$\boldsymbol{A}^{(i)} = \begin{cases} (\boldsymbol{I} - \eta \boldsymbol{D}^{(i)}) & i < d \\ \boldsymbol{I} & i = d, \end{cases} \tag{14}$$

with $\eta$ being a parameter of choice and $\boldsymbol{I}$ being an identity matrix of appropriate dimensions. The diagonal matrix consisting of the scaling factor is subtracted from an identity matrix to ensure that a large decay is introduced in the rows with smaller singular values. A small decay is introduced where larger singular values are observed. With $\boldsymbol{A}^{(i)}$ as defined in Eq. (14), the layer-wise cost function is modified as

$$H^{(i)}(\hat{\boldsymbol{W}}^{(i)}) = \frac{1}{2}\left[tr((\boldsymbol{\delta}_k^{(i)})^T \boldsymbol{P} \hat{\boldsymbol{W}}_k^{(i)}) + \lambda tr((\boldsymbol{W}_k^{(i)})^T \boldsymbol{A}^{(i)} \boldsymbol{W}_k^{(i)})\right],$$

which results in the layer-wise update rule with $\boldsymbol{\delta}_k^{(i)}$ as defined in Eq. (12). These update rules are summarized in Table 1. A few implementation related remarks are discussed next.

**Remark 19** *1. For each batch of data, an alternate batch representing the neighborhood is generated can be seen in Figure 1b. The larger the size of the alternate batch, the better is the approximation of the neighborhood. As part of our simulations, for each data-point in a batch, two additional data-points are generated, which worked well in practice. Sampling methodologies such as bootstrap [32] may be utilized for efficient generation of the alternate batches.*

2. *The learning methodology progresses batch-wise where random sampling is used to determine batches in classification. Therefore, for every update, expected values are evaluated over the batch, assuming uniform distribution.*

Table 1. Summary of the batch-wise updates with $\boldsymbol{P}$ chosen as an identity matrix.

| | $(\boldsymbol{W}^{(i)})$ |
|---|---|
| Gradient descent | $\boldsymbol{u}_k^{(i)} = E[\boldsymbol{\delta}_k^{(i)}] + \lambda \boldsymbol{W}_k^{(1)}$, where $\boldsymbol{\delta}_k^{(i)}$ is defined in Eq. (10). |
| EDL | $\boldsymbol{u}_k^{(i)} = E[\boldsymbol{\delta}_k^{(i)}] + \lambda \boldsymbol{A}^{(i)} \boldsymbol{W}_k^{(i)}$, where $\boldsymbol{\delta}_k^{(i)}$ is defined in Eq. (12). |

The proposed methodology is tested on several data-sets and the results are presented in the next section.

## 4. RESULTS AND DISCUSSION

A total of ten data-sets are used for analysis and the statistics for these data-sets are summarized in Table 3. In all of the data-sets considered here, 80% of the available data is chosen for learning and twenty percent is used to test the trained model [24]. All of the results demonstrated in the next section are calculated from the data used to test the trained model. The splits are constructed randomly.

For the proposed framework, Gaussian-distributed and uniformly-distributed perturbations are introduced during the learning phase to construct the neighborhood. We train the NN with the mean vector for the Gaussian distribution, chosen as a zero vector, whereas the variance matrix for the perturbations is chosen as $\rho \boldsymbol{I}$, with $\boldsymbol{I}$ being an appropriate identity matrix and $\rho$ being the variance of choice. The parameters for the uniformly-distributed noise are chosen as $[-\rho, \rho]$. One may enforce the size of the neighborhood by writing $\Delta \boldsymbol{x} = \frac{\Delta \boldsymbol{x}}{\|\Delta \boldsymbol{x}\|}\rho$. The choice of $\rho$ during the learning phase represents the size of the neighborhood defined by the practitioner.

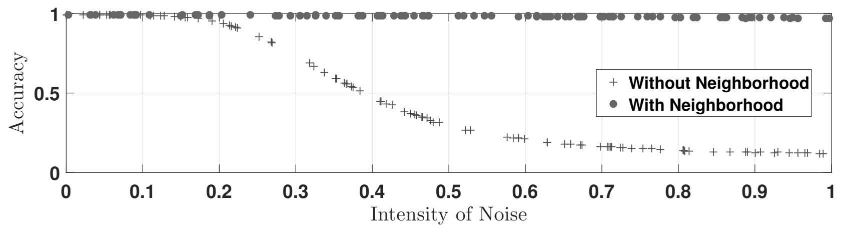
Table 2. Parameters of the different models used in this paper.

| Architecture | Parameters |
|---|---|
| CNN | $n_{conv} = 2$, $n_{layers} = 2$,<br>filter size = 5, pool shape = (3,3),<br>hln= 128, $\alpha = 0.01$, $b = 100$, $\lambda = 0.001$<br>$n_{neigh} = 2$, $\rho = 1$ |
| MLNN | $d = 7$, hln= 128, $\alpha = 0.01$, $b = 100$, $\lambda = 0.001$<br>$n_{neigh} = 2$, $\rho = 1$ |
| SAE | $n_{units} = 3$,<br>hln= 128, $\alpha = 0.01$, $b = 100$, $\lambda = 0.001$<br>$n_{neigh} = 2$, $\rho = 1$ |
| DAE | $n_{units} = 3$,<br>hln= 128, $\alpha = 0.01$, $b = 100$, $\lambda = 0.001$<br>$n_{neigh} = 2$, $\rho = 1$ |

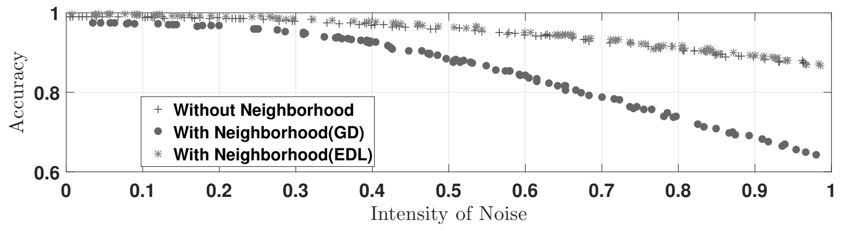
In all the simulations for EDL, $\boldsymbol{B}^{(i)}$ is chosen using the singular vectors of $\mathcal{T}^{(i)}$ at every layer. All of the results in this section are averaged over 100 initial conditions. The software package Tensor-flow with Python is used for all of the experiments in this paper. The robustness of the proposed framework to heterogeneity and data-noise for the proposed methodology on the MNIST data-set is studied first. The hyper-parameters are summarized in Tables 3 and 2. In Tables 2 and 3, hln refers to the number of hidden layer neurons at each layer; b denotes the batch size and $n_{neigh}$ represents the number of data-points in the neighborhood corresponding to each data-point in the data-set. Also, $n_{conv}$ represents the number of convolutional layer and $n_{layers}$ represents the number of feed-forward layers.

## 4.1. MNIST DATA-SET

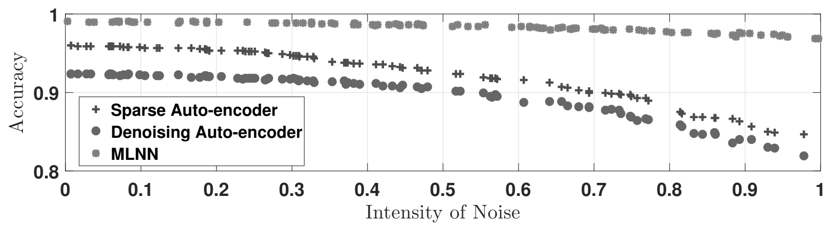
The impact of heterogeneity and data-noise is studied using generalization error to quantify the performance of the methodology. To simulate the presence of heterogeneity and data-noise in this data-set, randomly sampled perturbations are introduced into the data during the testing phase. The NN in the proposed framework is considered with tanh and relu activation functions, respectively. For the first set of results, we evaluate the methodology when the amount of noise in the data increases and the results are shown in Figure 2b. The



(a) CNN



(b) MLNN



(c) SAE and DAE

Figure 2. Change in the size of the neighborhood versus (top) accuracy (bottom) generalization error. Results presented in these plots are averaged over two types of perturbations that are Gaussian and Uniform distributed.

generalization performance of the proposed framework is studied for increasing dissimilarity of the test data introduced using synthetic perturbations. The neighborhood size for training is kept as one and the average accuracies and generalization error over one hundred initial conditions of weights are illustrated in Figure 2b. The deterioration in performance for the proposed framework with an increase in the perturbations is smaller with the proposed framework. Both learning regimes, EDL and GD, show improved performance. In Figure 2b, SGD results in accuracy that is a little higher than that achieved by EDL when no dissimilarity is present. However, EDL is very resilient to increasing dissimilarity in the data. In other words, the proposed approach, while exhibiting comparable best case accuracy, indicates improved resilience in the presence of data-noise and heterogeneity.

Next, we test the performance of convolutional neural networks (CNN) [33] and multilayer neural networks (MLNN), with respect to the proposed methodology. The hyper-parameters corresponding to these architectures are summarized in Table 2. For both CNN and MLNN, it is consistently observed that a significant improvement in performance is achieved when uniformly and Gaussian distributed perturbations are introduced to construct the neighborhood.

Next, the performance of the proposed methodology is compared with sparse auto-encoders [34] and denoising auto-encoders [35]. The hyper-parameters are again summarized in Table 2. The results are demonstrated in Figure 2c. The best case accuracies for varying amount of noise in the data-set is achieved by the proposed approach. The proposed methodology is built to address scenarios where the test data-set is different from the training data-set, whereas sparse auto-encoders and denoising auto-encoders are constructed to look for and remove noisy patterns in the training data-set. Moreover, explicit modification of the objective function appears to help and thus improve performance within the proposed approach.

Next, the performance of the proposed framework is studied relative to the regular SGD for the MNIST data-set, the results are summarized in Table 6. The accuracies are higher for relu compared to tanh activation functions. The improved performance of the relu activation function over tanh is well recorded in the literature [13]. The standard error of accuracy for the proposed framework is much smaller compared to traditional SGD. Therefore, learning with the proposed framework is resilient towards varied initial conditions and the results are statistically significant.

Next, with the use of proposed decay, classification is performed on the MNIST data-set, while synthetically introducing noisy dimensions. The noisy dimensions are sampled from a Gaussian distribution with known mean and standard deviation. The coefficient of decay is chosen to be 0.01. The results illustrated in Figure 3 demonstrates the performance of EDL with respect to a change in the number of noisy dimensions in the data. The change

Table 3. % accuracies for the various data-sets with the proposed framework and four different learning algorithms with generalization error in the parenthesis.

| Datasets | $p$ | $n$ | $\mathcal{F}$ | Model Parameters | **DFA** | **FA** | **SGD** | **EDL** |
|---|---|---|---|---|---|---|---|---|
| **Rolling** [36] | 11 | 35000 | 4 | $\alpha = 0.01, b = 100, d = 7, hln = 100,$ $\lambda = 0.001, n_{neigh} = 2, \rho = 1$ | 99(0.0001) | 99(0.0001) | 99(0.6) | 99(0.0) |
| **Sensorless** [37] | 48 | 78000 | 11 | $\alpha = 0.01, b = 100, d = 8, hln = 100,$ $\lambda = 0.001, n_{neigh} = 2, \rho = 1$ | 94(0.008) | 94(0.008) | 95(0.008) | 94(0.002) |
| **MNIST** [38] | 784 | 72000 | 10 | $\alpha = 0.01, b = 100, d = 8, hln = 100,$ $\lambda = 0.001, n_{neigh} = 2, \rho = 1$ | 92(0.1) | 94(0.2) | 95(1) | 97(0.6) |
| **NotMnist** [39] | 784 | 81000 | 10 | $\alpha = 0.01, b = 100, d = 9, hln = 100$ $\lambda = 0.001, n_{neigh} = 2, \rho = 1$ | 90(1) | 82(1.8) | 88(2) | 92(0.8) |
| **CIFAR10** [40] | 3072 | 50000 | 10 | $\alpha = 0.01, b = 100, d = 10, hln = 100$ $\lambda = 0.001, n_{neigh} = 2, \rho = 1$ | 81(7.21) | 80(8.00) | 82(12.11) | 84(8.77) |
| **Arcene** [41] | 10000 | 100 | 2 | $\alpha = 0.01, b = 100, d = 9, hln = 100, \lambda = 0.001$ $\lambda = 0.001, n_{neigh} = 2, \rho = 1$ | 69(0) | 61(0) | 51(0) | 78(0) |
| **Dexter** [41] | 20000 | 300 | 2 | $\alpha = 0.01, b = 100, d = 8, hln = 100$ $\lambda = 0.001, n_{neigh} = 2, \rho = 1$ | 71(0.7) | 77(0.9) | 81(1.1) | 80(1) |
| **Gisette** [41] | 5000 | 6000 | 2 | $\alpha = 0.01, b = 100, d = 8, hln = 100,$ $\lambda = 0.001, n_{neigh} = 2, \rho = 1$ | 92(0.25) | 93(0.21) | 91(0.3) | 98(0.07) |

in accuracy for EDL is smaller relative to SGD as seen in Figure 3. The above-mentioned results indicate that the proposed update law remains robust in the face of noisy dimensions in the data. The accuracies plateau when the number of dimensions increase beyond a certain limit. The behavior indicates that the use of proposed decay is able to demonstrate control over how the number of dimensions impact the performance.

Next, the performance of the proposed method is tested on classification.

## 4.2. CLASSIFICATION

Four learning paradigms, namely DFA (Direct Feedback Alignment) [18], SGD (Stochastic Gradient Descent) [4], FA (Feedback Alignment) [30] and EDL (Error-driven Learning), are tested with all the data-sets listed in Table 3. The hyper-parameters for different models used in the paper are summarized in Table 3. Consistent performance of the proposed framework with EDL is observed across data-sets as is seen in Table 3. Reasonable accuracies are observed for the large dimensional data-sets and others provide marginal improvement for typical SGD. For the dexter data-set, our EDL method achieved similar accuracies compared to standard SGD but indicates significant improvement over

Table 4. Computational time for training the NN over the synthetic data-set for 100 runs, where $\tau$ denotes the computational time.

| $n_{neigh}$ | $\tau$ (hours) |
|---|---|
| 0 | 88 |
| 2 | 90 |
| 3 | 97 |
| 5 | 108 |

Table 5. Average accuracies.

| Data-set | $n$ | $p$ | $\mathcal{F}$ | Accuracy |
|---|---|---|---|---|
| Synthetic | 20000 | 2 million | 10 | 94.10 |
| Image-Net[42] | 784 | 12614060 | 10 | 56.21 |

DFA and FA. The guided choice of the feedback matrix appears to assist EDL. Next, the generalization performance of the methodology for different data-sets is presented in Table 3. It is observed that the lowest generalization error is achieved for the proposed framework with the EDL learning scheme. Even for the case of gradient-based updates on the proposed framework, there is a significant reduction in the generalization error over the traditional SGD. Improved generalization errors indicate that the impact of challenges such as heterogeneity and data-noise is indeed mitigated with the proposed framework.

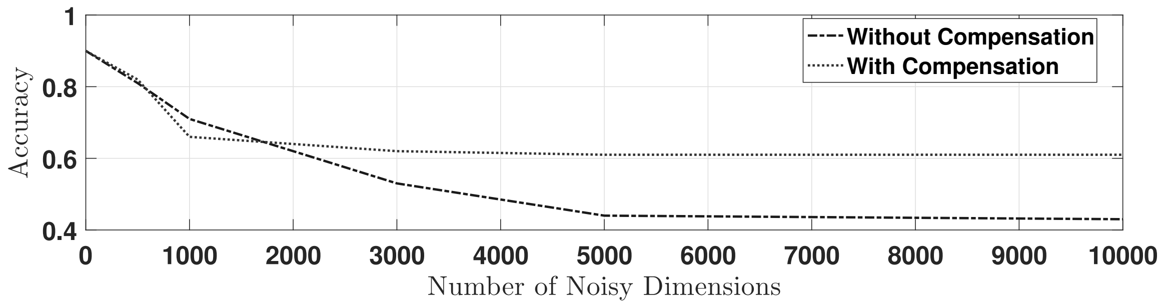


Figure 3. Accuracy with respect to increase in the number of dimensions in the data with decay.

Table 6. Mean test accuracies with the standard error in accuracy for the MNIST data-set.

| | **SGD** | **PF+GD** |
|---|---|---|
| relu (Gaussian) | 0.83(0.07) | 0.91(0.093) |
| relu (Uniform) | 0.85(0.10) | 0.93(0.018) |
| tanh (Gaussian) | 0.82(0.006) | 0.87(0.003) |
| tanh (Uniform) | 0.86(0.10) | 0.89(0.018) |

## 4.3. BIG-DATA APPLICATION

Big data is commonly defined as the scenario where the analysis of a big data-set cannot be conducted on a single core CPU. Therefore, to demonstrate the use of proposed methodology in such an environment, our approach is implemented on a multi-core CPU. The results are demonstrated on two large data-sets where the first is the popular ImageNet data-set [42], and the second is the synthetic data-set.

We construct a ten class problem from the ImageNet data-set by sampling with replacement. In the end, the data-set is about 138 GB in size consisting of ten classes. Similarly, the synthetic data-set is composed of 20,000 features and 2 million data-points. While generating this data-set, clusters of normally distributed points are created about vertices of an 20,000-dimensional hypercube with sides of a pre-defined length. Both of these data-sets are extremely large. The results, as summarized in Table 5, indicate the ability of the methodology to obtain reasonable performance.

## 4.4. COMPUTATIONAL OVERHEAD

Finally, since the amount of computational resources required for our methodology depends on the number of data-points that constitute the neighborhood, we study the increase in computational time with an increase in the number of data-points. The results are summarized in Table 4. Although there is an increase in the amount of time it takes for implementation, we observe that the increase is not substantial and justifies the improvement in the accuracy.

## 5. CONCLUSIONS

A systematic approach was presented to design a classifier in the presence of challenges such as heterogeneity, vanishing gradients and noise. Since synthetic distortions are introduced to approximate the generalization error, our framework is robust in the presence of data-noise because less variation in accuracy is observed. Careful selection of the magnitude of the perturbation is necessary because large perturbations deteriorate the performance of the proposed framework significantly.

The direct use of the overall error signal in the learning process appears to mitigate the vanishing gradient issue even when sigmoid and tanh activation functions were used. The proposed framework combined with EDL appears to outperform other approaches such as DFA and FA in all of the data-sets under analysis. Moreover, noisy dimensions appear to impact accuracy less with the use of the singular value-based decay in the proposed update law. Overall, the proposed approach appears to provide a 5% improvement in generalization error and a 10% improvement in accuracy over SGD in the presence of noise. Future effort will address the challenges such as non-stationary distributions and incidental endogeneity.

## ACKNOWLEDGMENT

This research was supported in part by NSF I/UCRC award IIP 1134721 and Intelligent Systems Center.

# REFERENCES

[1] J. Fan, F. Han, and H. Liu, “Challenges of big data analysis,” *National science review*, vol. 1, no. 2, pp. 293–314, 2014.

[2] J. Fan and Y. Liao, “Endogeneity in high dimensions,” *Annals of statistics*, vol. 42, no. 3, p. 872, 2014.

[3] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[4] M. Hardt, “3.12 train faster, generalize better: Stability of stochastic gradient descent,” *Mathematical and Computational Foundations of Learning Theory*, p. 64, 2015.

[5] X.-W. Chen and X. Lin, “Big data deep learning: challenges and perspectives,” *IEEE access*, vol. 2, pp. 514–525, 2014.

[6] K. Sun, S. H. Huang, D. S. H. Wong, and S. S. Jang, “Design and application of a variable selection method for multilayer perceptron neural network with lasso,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 6, pp. 1386–1396, June 2017.

[7] D. Mishkin and J. Matas, “All you need is a good init,” *arXiv preprint arXiv:1511.06422*, 2015.

[8] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting.” *Journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[9] N. Xu, J. Hong, and T. C. Fisher, “Generalization error minimization: a new approach to model evaluation and selection with an application to penalized regression,” *arXiv preprint arXiv:1610.05448*, 2016.

[10] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” *arXiv preprint arXiv:1412.6572*, 2014.

[11] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, p. 436, 2015.

[12] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training recurrent neural networks,” in *International Conference on Machine Learning*, 2013, pp. 1310–1318.

[13] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[14] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[16] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[17] D.-H. Lee, S. Zhang, A. Fischer, and Y. Bengio, "Difference target propagation," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2015, pp. 498–515.

[18] A. Nøkland, "Direct feedback alignment provides learning in deep neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 1037–1045.

[19] J. Goldberger and E. Ben-Reuven, "Training deep neural-networks using a noise adaptation layer," 2016.

[20] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich, "Training deep neural networks on noisy labels with bootstrapping," *arXiv preprint arXiv:1412.6596*, 2014.

[21] Z. Yu, L. Li, J. Liu, and G. Han, "Hybrid adaptive classifier ensemble," *IEEE Transactions on Cybernetics*, vol. 45, no. 2, pp. 177–190, Feb 2015.

[22] F. Lewis, S. Jagannathan, and A. Yesildirak, *Neural network control of robot manipulators and non-linear systems*. CRC Press, 1998.

[23] S. Sonoda and N. Murata, "Neural network with unbounded activation functions is universal approximator," *Applied and Computational Harmonic Analysis*, 5 2015.

[24] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[25] P. Niyogi and F. Girosi, "On the relationship between generalization error, hypothesis complexity, and sample complexity for radial basis functions," *Neural Computation*, vol. 8, no. 4, pp. 819–842, 1996.

[26] Q. Song and F. Liang, "A split-and-merge bayesian variable selection approach for ultrahigh dimensional regression," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 77, no. 5, pp. 947–972, 2015.

[27] J. Pearl, "Detecting latent heterogeneity," *Sociological Methods & Research*, vol. 46, no. 3, pp. 370–389, 2017.

[28] Z. Wu, C. Shen, and A. v. d. Hengel, "Wider or deeper: Revisiting the resnet model for visual recognition," *arXiv preprint arXiv:1611.10080*, 2016.

[29] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks." in *CVPR*, vol. 1, no. 2, 2017, p. 3.

[30] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman, "Random synaptic feedback weights support error backpropagation for deep learning," *Nature communications*, vol. 7, p. 13276, 2016.

[31] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM review*, vol. 38, no. 1, pp. 49–95, 1996.

[32] P. Hall, *The bootstrap and Edgeworth expansion*. Springer Science & Business Media, 2013.

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[34] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of machine learning research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[35] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *Advances in neural information processing systems*, 2012, pp. 341–349.

[36] A. Soylemezoglu, S. Jagannathan, and C. Saygin, "Mahalanobis taguchi system (mts) as a prognostics tool for rolling element bearing failures," *Journal of Manufacturing Science and Engineering*, vol. 132, no. 5, p. 051014, 2010.

[37] M. Lichman, "UCI machine learning repository," 2013. [Online].

[38] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[39] Y. Bulatov, "Notmnist dataset," *Google (Books/OCR), Tech. Rep.[Online].*, 2011.

[40] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[41] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror, "Result analysis of the nips 2003 feature selection challenge," in *Advances in neural information processing systems*, 2005, pp. 545–552.

[42] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[43] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," *arXiv preprint arXiv:1609.04836*, 2016.

# IV. A GAME THEORETIC APPROACH FOR ADDRESSING DOMAIN-SHIFT WITH APPLICATIONS TO BIG-DATA CLASSIFICATION

R. Krishnan, S. Jagannathan

Department of Electrical & Computer Engineering

Missouri University of Science and Technology

Rolla, Missouri 65409–0050

Email: krm9c, sarangap@mst.edu

V.A. Samaranayake

Department of Mathematics & Statistics

Missouri University of Science and Technology

Rolla, Missouri 65409–0050

Email: vsam@mst.edu

## ABSTRACT

In this paper, a novel approach is presented to mitigate the issue of domain shift in big data classification. In the proposed approach, a suitably selected domain shift is first introduced to the training data via a "distortion model" and additional data-samples are obtained. Next, a deep neural network, referred as "classifier," is used to classify both the training data and the additional data samples. Since the additional data-samples represent the domain shifted data, the classifier can compensate for domain shift while maintaining its performance on the training data. Furthermore, as the exact magnitude of this shift is unknown a priori, in this work, we compensate for the optimal shift by formulating a zero-sum game. In the proposed game, the distortion model is viewed as the maximizing player which increases the domain shift while the classifier becomes the minimizing player. The

Nash solution of the game provides the domain shift and its optimal adaptation through the classifier. To solve the proposed game for the Nash solution, a direct error-driven learning scheme is introduced. A comprehensive simulation study and mathematical analysis is presented to demonstrate the efficiency.

## 1. INTRODUCTION

Generally, when deep neural networks (DNN) are used for extracting insights from big data, the underlying distribution generating the data is assumed stationary [1]. In other-words, the data-distribution in the learning phase, "source distribution," is assumed similar to that in the testing phase, "target distribution." However, this is untrue in many practical real world scenarios [1]. Consider an example where defects in a roller bearing are to be detected. Let the source distribution involve a data-set with no motor load. If the bearing system comes under different motor load states in the testing phase, the data distribution will not be similar. Learning therefore involves compensating for a shift in the data-distribution [2]. The phenomenon is called domain shift (DS) while the compensation of this shift is referred to as domain adaptation (DA) or transductive transfer learning [2].

DS can be observed due to a number of reasons such as data-noise and statistical heterogeneity [3]. Since this shift is often unpredictable and/or unknown, it introduces errors, known as generalization errors, in the data analytics including prediction. To counter this problem, a novel methodology is introduced in this paper to model and reduce generalization errors specifically due to the challenges mentioned above. The problem under consideration is that of classification.

In the literature, norm regularization [4], dropouts [5] and data-augmentation [6] have been utilized to reduce the sensitivity of DNN towards distribution changes. Furthermore, several DNN based approaches have been proposed to compensate for DS [7, 8].

The afore-mentioned approaches [4, 5, 6, 7, 8], suffer from two major drawbacks: (1) the performance depends on the good use of heuristics [4, 5, 6], which is difficult to obtain in the big data environments and (2) they work well under a general assumption: the source and the target distributions are known and available [8, 7], an impractical assumption in many engineering environments.

To counter the problem, additional data-samples that are representative of the unknown distribution shift, referred as "$\psi_T$," are generated and the deep NN is used to classify these samples. To generate these samples, the unknown shift is simulated by introducing variations in the source distribution ($\psi_S$) through a distortion model under the assumption that the shift can be modeled by a suitably selected transformation. The additional data-samples provide an estimate of the target distribution, referred as "$\psi_T$."

To learn the parameters of the distortion model and the classifier, an overall cost is defined to include the cost due to learning and generalization errors and a measure of DS introduced by the distortion model. The true targets for $\psi_S$, and $\psi_S$ provide an estimate of learning error. Moreover, $\psi_T$ and the corresponding targets provide an estimate of the generalization error. A measure of DS is then obtained by calculating the difference between $\psi_S$ and $\psi_T$.

Since the targets corresponding to $\psi_T$ are unknown, a nearest neighbor based approach is introduced to estimate them. In this approach, for every data-point $x$ in $\psi_T$, the distance between the classifier output of $x$ and that of all the data-points in $\psi_S$ is first evaluated. Then, depending on the magnitude of the distance values, the neighbors of the classifier output of $x$ are determined. The final label for $x$ is assigned based on the true targets for these neighbors from $\psi_S$. The final objective is to minimize the overall cost while introducing optimal DS through the distortion model.

Since, these are conflicting objectives, a two player zero-sum game is designed where distortion model is treated as the first player whose aim is to increase the DS. On the other hand, the second player is the classifier where its aim is to minimize the impact of DS for adaptation. In the proposed game, the distortion model manipulates the data to introduce variations in the output of the classifier thus introducing generalization error. On the other hand the classifier compensates for the change in generalization error through adaptation of weights. The Nash solution of the game provides the DS and its optimal adaptation through the classifier.

In traditional games, system dynamics play a key role in deciding the solution of the two player zero-sum game because each player is a dynamical system. Furthermore, majority of results present in the literature support such an interplay [9]. In contrast, in this work, no dynamical systems are present and the dynamics for the game are introduced by the parameter updates for the two players.

To solve the proposed game, a direct error-driven learning scheme is introduced where a combination of the learning and generalization error is multiplied by a projection matrix to tune the classifier. On the other hand, a measure of DS is multiplied by a projection matrix to train the distortion model. To enable this learning, a cost function is defined at each layer in the deep NN as a function of overall cost. Finally, the projection matrix and the weights at each layer are optimized to minimize the sum of all the layer-wise costs. The proposed learning rule is generic enough to solve any cost function. As an added benefit, the use of the projection matrix to directly utilize the error for learning mitigates the issue of vanishing gradients in contrast with traditional learning schemes such as SGD [10]. Direct error-driven learning was first introduced in [11]. Despite its demonstrated effectiveness, the main drawback was that the projection matrix had to be chosen manually. In contrast with [11], the projection matrix is optimized in the learning regime proposed in this paper.

Finally, theoretical analysis is presented. Specifically, we show that the saddle point solution of the two player game exists. Furthermore, we quantify the changes introduced by the choice of the distortion model on the first two moments of the data-distribution. The benefits of the proposed approach are demonstrated on a total of ten data-sets, a few of which fall into the category of big data. The contributions of this paper include:

Table 1. Notations

| **Notation** | **Meaning** |
|---|---|
| $\boldsymbol{\mathcal{X}}$ | Data-set |
| $\mathcal{F}$ | Number of faults/classes in the system. |
| $(n, p)$ | Number of sample points in the training set, the dimensionality of data |
| $\mathbb{R}$ and $\mathbb{N}$ | The set of real and natural numbers. |
| $(.)^{(i)}$ | Number of layers in the NN |
| $\boldsymbol{W}, \hat{\boldsymbol{W}}$ | Ideal and estimated weights |
| $k, t$ | Iteration index in the learning phase and the test phase |
| $\varepsilon$ | The approximation error |
| $f^{(i)}, for \quad i = 1 \cdots d$ | layer-wise activation functions with $d$ layers |
| $\psi_S$ and $\psi_T$ | The source and the target distribution |

1. A generic distortion model for DA in the presence of big data challenges such as heterogeneity and noise. Our approach estimates the DS using a neighborhood when the target distribution is unknown in contrast with the current literature [4, 6, 12, 8, 7]. Further, compensation for higher order changes in the data-distribution is possible in contrast with [11], where only mean shift is considered.

2. Design of a two player zero-sum game for classification. In the proposed approach, the estimation of the DS is part of the learning problem in contrast with [4, 6, 12, 8, 7].

3. A direct error driven alternating minimization approach for solving the zero-sum game. In contrast with traditional learning regimes such as back-propagation [10], the challenges from big-data are directly addressed during learning.

4. Simulation and mathematical analysis of the proposed approach involving a total of ten data-sets. The theoretical results in the paper are generalized to the case when the number of data-points, the number of dimensions, the number of layers in the NN and the number of neurons at each layer all tend to infinity.

## 2. PROBLEM STATEMENT

The basic notations for the paper are summarized in Table 1. The problem of classification can be divided into two phases, the learning and the testing phase. In the learning phase, the DNN learns from $\psi_S$, the "source distribution," and the iteration index is denoted as $k$. The second phase is called the testing phase where the DNN is used for testing the data-points from the "target distribution," denoted as $\psi_T$ and $t$ is used as the index.

Let $\boldsymbol{x}_k \in \psi_S$ represent a data-point at the learning instant $k$. Next, define the objective of learning as that of determining the class/category corresponding to $\boldsymbol{x}_k$ using an input-output map $\phi(.)$. Let there be a total of $\mathcal{F}$ classes/categories such that $\psi_S = \{\cup_{i=1}^{\mathcal{F}} \psi_i\}$.

Under the assumption that $\phi(.)$ can be approximated using a DNN with ideal weights denoted as $\boldsymbol{\theta} = [\boldsymbol{W}^{(1)} \cdots \boldsymbol{W}^{(d)}]^T$, one may write [13]

$$\phi(\boldsymbol{x}_k) = \boldsymbol{y}(\boldsymbol{x}_k;\theta) = f^{(d)}(\boldsymbol{W}^{(d)} \cdots (f^{(1)}(\boldsymbol{W}^{(1)}(\boldsymbol{x}_k))))) + \varepsilon, \tag{1}$$

where $\boldsymbol{y} \in \mathbb{R}^{\mathcal{F}\times 1}$ represents the ideal output for $\phi(.)$ and the target weights includes the layer-wise bias and there are $d$ layers in the DNN. Consider the estimated map be denoted as $\hat{\boldsymbol{y}}(\boldsymbol{x}_k)$ with weights given as $\hat{\boldsymbol{\theta}} = [\hat{\boldsymbol{W}}^{(1)} \cdots \hat{\boldsymbol{W}}^{(d)}]^T$ such that

$$\hat{\boldsymbol{y}}(\boldsymbol{x}_k) = \hat{\boldsymbol{y}}(\boldsymbol{x}_k;\hat{\boldsymbol{\theta}}) = f^{(d)}(\hat{\boldsymbol{W}}^{(d)} \cdots (f^{(1)}(\hat{\boldsymbol{W}}^{(1)}\boldsymbol{x}_k)))). \tag{2}$$

A sample from the source distribution is denoted as $\boldsymbol{\mathcal{X}}$ and is used for learning the DNN. The following assumptions are necessary before proceeding further.

**Assumption 8** *Samples from each fault/category are obtained such that they are independently and identically distributed.*

**Assumption 9** *The probability distribution for each of the categories in the training phase is similar to the distribution of the data in the testing phase.*

With these assumptions, the learning objective can be defined as the estimation of $\phi$ using a DNN. However, when the problem under consideration deals with the high dimensionality and volatility of big data, the second assumption is violated. In other words, a difference between $\psi_S$ and $\psi_T$ is observed. Furthermore, it is expensive or even impossible to recollect the required training samples for reconstructing the classifier such that we can account for the sudden change [14]. Therefore, it is preferable for the DNN to compensate for such changes in the learning phase, which is the aim of this paper. In the literature, such compensations exists and the related work is discussed next.

## 3. RELATED WORK

In the literature, several Deep NN-based approaches have been proposed to compensate for DS. Taigman et al. [7] proposed a domain transfer network (DTN), that allows to translate images from a source domain to a target one, under a f-constancy constraint, where f is a generic function that maps images in a feature space and the translated target images are very similar to the source domain data.

Similarly, a two neural network architecture is proposed in [19], where the weights are adapted to each domain independently and weight sharing is enabled to enforce collaboration between the two neural networks. Haeusser et al. [20] propose a similar two NN structure to exploit and optimize associations between source and target features during training. On the other hand in [21], the authors optimize the relationship between the target features and the decision boundary by adding a discrepancy loss into the problem.

In contrast with the approaches in [24, 4, 19, 5, 25, 12, 21, 20, 7], several adversarial training based approaches have been proposed too. Ganin and Lempitsky [8] propose a neural network (Domain-Adversarial Neural Network, DANN) where a ConvNet-based feature extractor is optimized to both correctly classify source samples and have domain-invariant features, through adversarial training. The main feature in [8] is the prediction of targets for the target distribution. Similarly, in Saito et al. [26] propose an approach where pseudo-targets are inferred and exploited for target samples during training. In particular, two networks are trained, one to assign targets to target samples and one to obtain target-discriminative features.

In contrast with the approaches in [27, 17, 28, 29, 8], the target distribution need not be known, but is approximated in the learning process with our work. In contrast to [27, 17, 28, 29], our approach performs feature learning and the estimates the target distribution in a unified architecture within the same learning phase and thus there is no need for hand-crafted features.

To improve the performance when target distribution was unknown, a methodology was introduced in [11], where target distribution was approximated. The approximation was derived under the assumption that the target distribution lies on the same manifold of the source distribution and can be approximated by defining a neighborhood around the source distribution. Although, this method was shown to work well, there are two drawbacks to this approach: it is difficult to predict the neighborhood. Secondly, the neighborhood was created by introducing additive perturbations into the training data-set. The associated drawback is that, one can only account for changes in the mean, that is the first moment, of the data-distribution which restricts the applicability. Furthermore, in [11], it is assumed that all the additional samples have the same label as that of the original data-point from $\psi_S$.

Similar to [11], a notion of neighborhood is utilized in this paper too. However, in contrast with [11], where additive perturbations are used to quantify the neighborhood, a generic transformation of the training set is introduced to quantify the neighborhood due to which, the proposed approach can compensate for changes in higher order moments. Moreover, in contrast with [11], the size of the neighborhood is estimated as part of the learning process instead of being fixed. To counter the problem of assumed labels, it is first shown that the assumption is impractical and then a nearest neighbor approach is proposed.

Furthermore, even in the case when the domain adaptation is not the goal, with the proposed framework, other discrepancies in the data distribution can be considered as part of the learning process thus increasing the applicability of the proposed approach in contrast with [27, 17, 28, 29, 15, 14, 16, 8] which are restricted to the field of domain adaptation.

The proposed approach has the following benefits: (1) the generic model is presented here to estimate DS which allows the flexibility to control and compensate for different types of distribution changes, (2) the NN exhibits better generalization ability and (3) it allows for compensation of shift even when the target distribution is unknown. The rest of the paper is organized as follows. Section II describes the proposed methodology. Section III describes the simulation results with Section IV providing the conclusions for the paper.

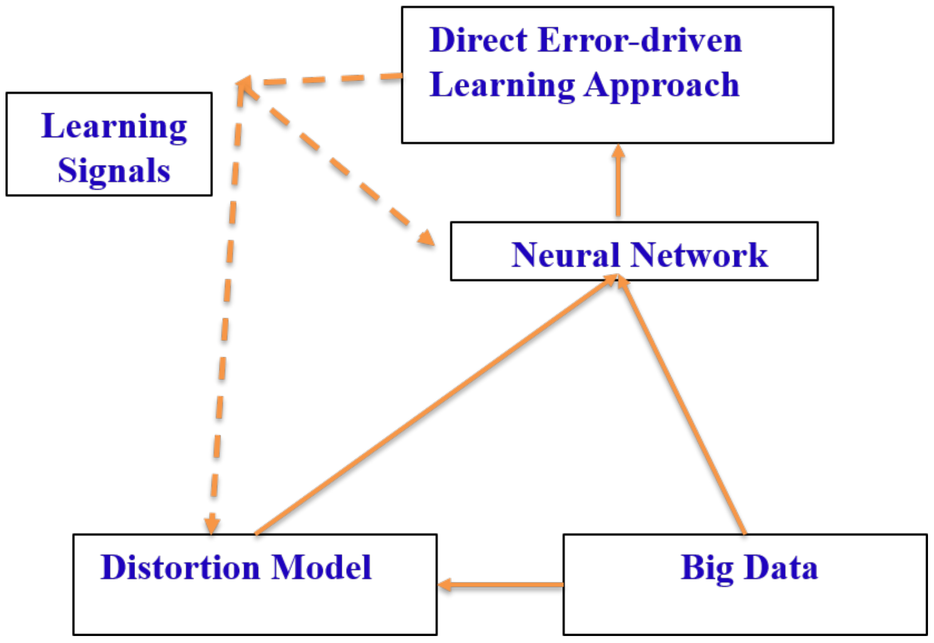


Figure 1. Overview of the Proposed Approach.

## 4. METHODOLOGY

The block diagram of the overall methodology is demonstrated in Figure 1. The main objective in this work is to compensate for a shift in the data-distribution in the learning phase. Since this shift is unknown and unpredictable, data-samples that represent

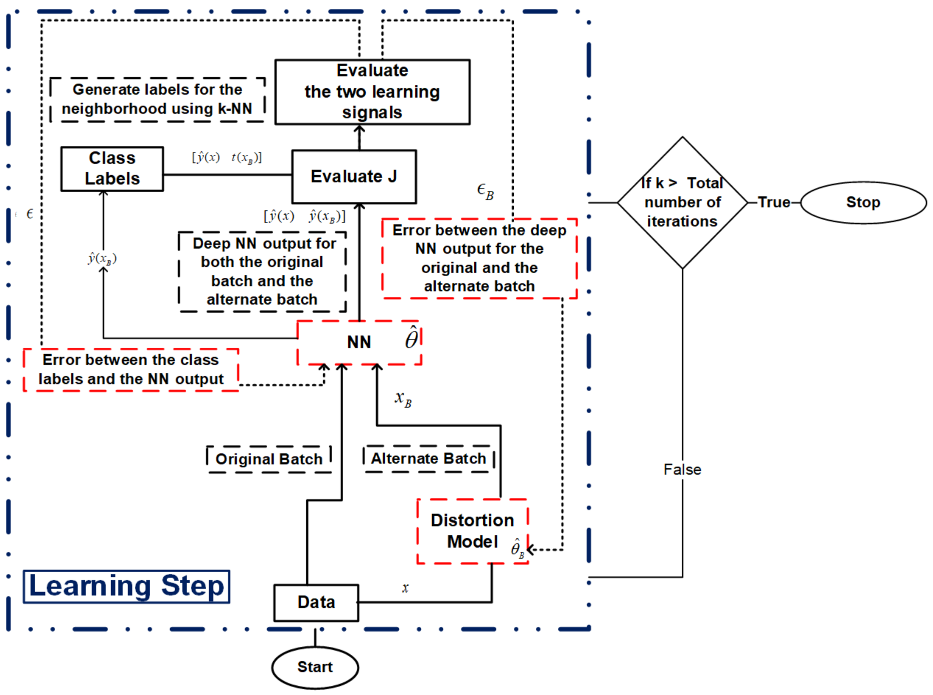

Evaluate the two learning signals
Generate labels for the neighborhood using k-NN
Class Labels
$[\hat{y}(x) \quad t(x_B)]$
Evaluate J
$\epsilon_B$
$\epsilon$
$[\hat{y}(x) \quad \hat{y}(x_B)]$
Deep NN output for both the original batch and the alternate batch
Error between the deep NN output for the original and the alternate batch
$\hat{y}(x_B)$
If k > Total number of iterations
True
Stop
NN $\hat{\theta}$
Error between the class labels and the NN output
$x_B$
Original Batch
Alternate Batch
False
Distortion Model $\hat{\theta}_B$
Learning Step
Data
$x$
Start

Figure 2. Block Diagram for the Proposed Approach.

the distribution shift are derived. The shift is introduced through the distortion model where a suitably selected transformation is utilized. The transformation is chosen according to the application under consideration.

Once these additional samples are generated, it is important to know the categories/class targets for these samples. A nearest neighbor based approach is employed to predict these targets. Finally, the classifier is used to learn on both the additional data-samples and the original data-samples.

The learning objective is to increase the extent of domain shift in the data and compensate for it through the classifier. However, as the extent of distribution shift must be obtained through the learning problem, we will design a two player zero sum game between the distortion model and the classifier. In the proposed game, the distortion model manipulates the data to introduce variations in the output of the classifier thus introducing generalization error. On the other hand the classifier compensates for the change in generalization error through adaptation of weights. To learn the proposed game, we will introduce a direct error driven learning regime.

The details are individually explained below where the learning objective is described first. Next, the construction of the distortion model and the label predictor is illustrated and afterwards, we describe the two-player zero-sum game and finish with the direct error-driven learning methodology.

## 4.1. LEARNING OBJECTIVE

At learning instant $k$, let the expected value of the error $\boldsymbol{e}(k)$ be defined as the difference between Eq. (1) and Eq. (2) such that

$$E[\boldsymbol{e}(k)] = E[\boldsymbol{e}_l(k)] + \varepsilon, \tag{3}$$

where $E[.]$ refers to the expectation operator and $E[\boldsymbol{e}_l(k)] = E_{\forall \boldsymbol{x} \in \boldsymbol{\chi}}[\boldsymbol{y}(k) - \hat{\boldsymbol{y}}(k)]$ is referred as learning error. The cost function to minimize is obtained by squaring the total error resulting in

$$\begin{aligned} J(k) &= E[\boldsymbol{e}(k)]^T E[\boldsymbol{e}(k)], \\ &= E[\boldsymbol{e}_l(k)]^T E[\boldsymbol{e}_l(k)] + 2\varepsilon^T E[\boldsymbol{e}_l(k)] + \varepsilon^T \varepsilon, \\ &= J_{emp}(k) + J_{gen}(k) + J_{apx}, \end{aligned}$$

with $J_{emp} = E[\boldsymbol{e}_l(k)]^T E[\boldsymbol{e}_l(k)]$ being the empirical cost and $J_{gen}(k) = 2\varepsilon^T E[\boldsymbol{e}_l(k)]$ and $J_{apx} = \varepsilon^T \varepsilon$ being the cost introduced by generalization and approximation errors, respectively [30]. Assuming that $J_{apx}$ is bounded [31], we will write $J(k) = J_{emp}(k) + J_{gen}(k)$. Formally, the learning objective is written as

$$\boldsymbol{\theta}^* = \arg min_{\hat{\theta} \in \Omega} J(k), \tag{4}$$

where $\Omega$ represents the parameter space. Note that due to the presence of big data challenges or DS, there is an increase in $J_{gen}(k)$, that is, an increase in generalization error. To generate additional samples that are representative of this increase, we will introduce the distortion model.

## 4.2. DISTORTION MODEL

To generate the additional samples, synthetic distortions are introduced into the data through a distortion model. Let this transformation be defined as $g(.;\hat{\boldsymbol{\theta}}_B)$, where $\hat{\boldsymbol{\theta}}_B$ denotes the parameters of the transformation such that $\psi_T = g(\psi_S; \hat{\boldsymbol{\theta}}_B)$.

**Remark 20** *The transformation may be chosen of a general nonlinear form defined as* $(\hat{\boldsymbol{\theta}}_B)^T \Psi(\boldsymbol{x})$. *Other choices of the transformation include affine map or a mean shift [11].*

Formally, for every $\boldsymbol{x} \in \boldsymbol{\mathcal{X}}$, there exists a

$$g(\boldsymbol{x}; \hat{\boldsymbol{\theta}}_B)$$

within the ball $Q_{\hat{\boldsymbol{\theta}}_B}$ with $\hat{\boldsymbol{\theta}}_B$ denotes the estimated parameters of the transformation. A collection of $g(\boldsymbol{x}; \hat{\boldsymbol{\theta}}_B)$ represents the neighborhood for $\boldsymbol{x}$ which results in $\boldsymbol{\mathcal{X}}_B = \{\boldsymbol{x}_B | \boldsymbol{x}_B = g(\boldsymbol{x}; \hat{\boldsymbol{\theta}}_B), \forall \boldsymbol{x} \in \boldsymbol{\mathcal{X}}\}$, where $\boldsymbol{\mathcal{X}}_B$ represents the target distribution. The distortion model is responsible for introducing domain shift, which is introduced due to the challenge of heterogeneity and noise. The choice of the distortion model is therefore important in determining the type of distribution changes that can be compensated through the use of distortion model. Under certain assumptions the distribution changes introduced by different choices of transformation can be quantified. These assumptions are as follows.

**Assumption 10** *The distribution of the* $\boldsymbol{X}$ *has an absolute continuous component.*

**Assumption 11** *The transformation* $g(.)$ *is bounded.*

Assumption 10 implies that a Taylor series expansion of the data-distribution can be derived and is satisfied by most data-distributions [32]. On the other hand, assumption 11 states that $g(.)$ is assumed to be a bounded transformation. It will be shown later in this paper that with the proposed learning methodology, the transformation remains bounded. The following lemma quantifies the distribution changes through the first two moments of the distorted data.

**Lemma 6** *Consider a data-sample of a random variable be given as $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ where $p$ denotes the number of dimensions in the data. Then, the moment generating function is given as*

$$\Psi(t) = 1 + tE[g(\boldsymbol{X})] + \frac{t^2}{2!}E[g(\boldsymbol{X})]^2 + \cdots. \tag{5}$$

*Furthermore, the first two moments when $n, p \rightarrow \infty$ can be quantified under two cases*

*Case 1: In a simple case, consider affine transformation $\boldsymbol{\mathcal{X}}$ and consider $\hat{\mathbf{A}}, \hat{\boldsymbol{b}}$ to be the parameters of the affine transformation*

$$\begin{aligned}||\mu_1|| &\leq \lambda_{max}(\hat{\boldsymbol{A}})||\mu(\boldsymbol{X})|| + ||\boldsymbol{b}||,\\ ||\mu_2|| &\leq \lambda_{max}(\boldsymbol{\mathcal{A}})\lambda_{max}(\boldsymbol{\Sigma}(\boldsymbol{X})) + ||\boldsymbol{b}||\\ &+[\lambda_{max}(\boldsymbol{\Sigma}(\boldsymbol{X})) + 2\lambda_{max}(\boldsymbol{A})||\boldsymbol{b}||]||\boldsymbol{\mu}(\boldsymbol{X})||,\end{aligned}$$

*where $\boldsymbol{\Sigma}(\boldsymbol{X}) = \boldsymbol{X}^T\boldsymbol{X}$ is the asymptotic covariance matrix and $\boldsymbol{\mathcal{A}} = \boldsymbol{A}\boldsymbol{A}^T$. Furthermore, $\boldsymbol{A}, \boldsymbol{b}$ are the parameters of the affine transformation and $\boldsymbol{\mu}(\boldsymbol{X})$ is the asymptotic mean.*

*Case 2: In a general case, let $g(\boldsymbol{X})$ be a generic nonlinear transformation such that $g(\boldsymbol{X}) = \hat{\boldsymbol{\theta}}_B^T\Phi(\boldsymbol{X})$. Following the same steps as before, we can conclude the first moment as*

$$\begin{aligned}||\mu_1|| &\leq ||\boldsymbol{\theta}_B||||\mu(\Phi(\boldsymbol{X}_j))||\\ ||\mu_2|| &\leq \lambda_{max}(\boldsymbol{\mathcal{A}})\lambda_{max}(\Sigma(\Phi(\boldsymbol{X}))),\end{aligned}$$

*where $\mathcal{A} = \boldsymbol{\theta}_B \boldsymbol{\theta}_B^T$ and $\lambda_{max}()$ represents the largest eigenvalue. Proof: Please see the appendix.*

***Remark 21*** *The lemma above requires the sample estimates of mean and covariance to converge to their population estimates. Law of large numbers guarantees this under the condition that $n, p \to \infty$ and $n \to \infty$ at a rate slower than $p$.*

***Remark 22*** *The covariance matrix would be full rank in the case when $n \to \infty$ at a rate slower than $p$. . However, when $p \to \infty$ at a rate equal to or greater than that of $n$, one would observe a sparse covariance matrix that may be singular.*

As observed, the upper limit of the moments converges as a function of the norm of the weight for the distortion model. Therefore, the transformation effects different moments on the data depending on the type of transformation that is applied on the source distribution. It stands to reason that depending on the type of transformation applied to generate this neighborhood, higher order changes in the data-distribution can be compensated for in the learning process.

In case of images when the problem of object detection is considered, an affine transformation could allow compensating for changes in object orientation, lighting etc. for the objects. On the other hand, when a generic nonlinear transformation is considered, the methodology can compensate for contraction and expansion of the data-distribution.

**Remark 23** *If more complicated changes in the data distribution are sought then parametric maps such as neural networks may be considered for choosing $g$. However in this case, the methodology will take the form of a generative adversarial network (GAN) with the key difference that the generator distorts the images instead of creating real images from random input [33].*

In summary, the transformation used in the distortion model must be chosen as per the application that is under consideration. To learn the classifier using these additional samples we need the targets corresponding to these additional samples.

Let $\boldsymbol{t}(\boldsymbol{x}_B)$ represent the targets for the data-points in the neighborhood. In a simple case, the targets for the neighborhood can be achieved by assuming that all the data-points in the neighborhood has the same class as that of the center point itself [11]. However, as the size of the neighborhood grows, one may observe a situation, where the neighborhood are overlapping. In such cases, the data-points in the overlapping areas might not belong to the same class as that of the center. In fact, these data-points might belong to a different class. In which case, the assumption is violated.

However, it is reasonable to assume that the data-points in the overlapping neighborhood belong to the same class as that of its closest spatial neighbor. In this case, it is of significance to determine the spatial location of data-points in the overlapping area relative to the center point. The spatial location can then provide the information corresponding to the target for that data-point which is core idea behind the proposed label detection approach as discussed next. Note that the idea of defining targets of the data based on their spatial location extends to the range space of the classifier too. Thus, we would talk about these targets in the range space of the classifier. That is, after the classifier function has been applied to the data.

Define, for every $\boldsymbol{x}_B \in \boldsymbol{\mathcal{X}}_B$, a vector $D = \{d(\hat{y}(\boldsymbol{x}_B), \hat{y}(\boldsymbol{x})), \forall \boldsymbol{x} \in \boldsymbol{\mathcal{X}}\}$, where $d(\hat{y}(\boldsymbol{x}_B), \hat{y}(\boldsymbol{x}))$ represents some distance metric. Next let $D$ be arranged in decreasing order. For the purpose of detection, we will determine what is the spatial location of the data-point in the neighborhood relative to the different data-points in $\psi_S$. Let $\zeta$ as the number of neighbors to be considered. Let $\hat{D}$ denote the first $\zeta$ elements in $D$, then the targets are finally given as $\boldsymbol{t}(\boldsymbol{x}_B) = \left[\frac{\sum_{i=1}^{\zeta} y(\boldsymbol{d}_i)}{\zeta}\right]$, where $[\quad]$ is the rounding off function and $y(\boldsymbol{d}_i)$ are the true targets corresponding to the $i^{th}$ elements in $\hat{D}$. Moreover, $\boldsymbol{d}_i$ is the corresponding neighbor in $\psi_S$. Traditionally, in nearest neighbor based approaches, the neighbors are evaluated in the data-space that is the domain of the classifier. In contrast, the neighbors are evaluated in the range-space of the classifier which is in the space of outputs of the classifier. Observe that the neighbors are determining based on the range space for the NN.

The approximation of generalization error that is obtained using $\psi_S$ and the targets corresponding to it can then be introduced into the cost function such that the distribution shift can be compensated for. However, the extent of the distribution shift or the size of the neighborhood is still unknown. These quantities are estimated as part of the learning problem that is designed next.

## 4.3. TWO PLAYER ZERO-SUM GAME

In the proposed game, the data is distorted such that it is reflected in an increased generalization error. This increase is approximated and optimized through the learning problem.

Formally, let the approximate generalization error be denoted as $\boldsymbol{e}_{gen}$ and write $E[\boldsymbol{e}_{gen}] = E_{\forall \boldsymbol{x}_B \in \boldsymbol{\chi}_B}[\boldsymbol{t}(\boldsymbol{x}_B) - \hat{\boldsymbol{y}}(\boldsymbol{x}_B; \hat{\boldsymbol{\theta}})]$ with the associated cost given as $\hat{J}_{gen} = E[\boldsymbol{e}_{gen}]^T E[\boldsymbol{e}_{gen}]$. The cost function in Eq. (4) is given as

$$J(k) = \quad J_{emp}(k) + \hat{J}_{gen}(k). \tag{6}$$

Note that, if the impact of $\hat{J}_{gen}(k)$ is larger on $J(k)$ relative to $J_{emp}(k)$ then an undesirable phenomenon is observed. The classifier performance would start improving on $\psi_T$ so much that the performance would significantly degrade on $\psi_S$ [34]. This phenomenon can be avoided by controlling the contribution of $J_{gen}(k)$ on the overall cost function. A parameter $\alpha$ is introduced into the cost function to effect this control such that

$$J(k) = \quad \alpha J_{emp}(k) + (1 - \alpha)\hat{J}_{gen}(k). \tag{7}$$

The approximated generalization cost is introduced into the learning problem at the training instant $k$ by rewriting Eq. (7) as

$$\begin{aligned} J &= \alpha E[\boldsymbol{e}_l(k)]^T E[\boldsymbol{e}_l(k)] + (1-\alpha)E[\boldsymbol{e}_{gen}(k)]^T E[\boldsymbol{e}_{gen}(k)] \\ &= [\boldsymbol{e}_l(k) \quad \boldsymbol{e}_{gen}(k)]^T \begin{bmatrix} \alpha & 0 \\ 0 & 1-\alpha \end{bmatrix} [\boldsymbol{e}_l(k) \quad \boldsymbol{e}_{gen}(k)] \\ &= \boldsymbol{\epsilon}(k)^T \boldsymbol{P} \boldsymbol{\epsilon}(k), \end{aligned}$$

where $\boldsymbol{\epsilon}(k) = [\boldsymbol{e}_l(k) \quad \boldsymbol{e}_{gen}(k)]$ denotes the overall error and $\boldsymbol{P} = \begin{bmatrix} \alpha & 0 \\ 0 & 1-\alpha \end{bmatrix}$. Observe that the expectation operators are dropped for brevity of notations and the iteration index $k$ is not explicitly stated from hereon. By minimizing $J$, we minimize the overall error which in-turn leads to the minimization of the learning error as well as the generalization error.

Next, define $d(\hat{\boldsymbol{y}}(\boldsymbol{x}), \hat{\boldsymbol{y}}(\boldsymbol{x}_B))$, as the measure of discrepancy between the NN output of the source and the neighborhood. In other words, $d(\hat{\boldsymbol{y}}(\boldsymbol{x}), \hat{\boldsymbol{y}}(\boldsymbol{x}_B))$, is a measure of the impact of neighborhood on the generalization error. Therefore, increasing $d(\hat{\boldsymbol{y}}(\boldsymbol{x}), \hat{\boldsymbol{y}}(\boldsymbol{x}_B))$ manipulates the distortion such that there is an increase in generalization error that is reflected in the output of the NN. Furthermore $d(.)$ is some distance metric where Mahalanobis distance or the Euclidean distance are possible choices.

Numerically, a ratio between the overall error $\boldsymbol{\epsilon}$ and $d(\hat{\boldsymbol{y}}(\boldsymbol{x}), \hat{\boldsymbol{y}}(\boldsymbol{x}_B))$ is defined as

$$\frac{\boldsymbol{\epsilon}(k)^T \boldsymbol{P} \boldsymbol{\epsilon}(k)}{d(\hat{\boldsymbol{y}}(\boldsymbol{x}), \hat{\boldsymbol{y}}(\boldsymbol{x}_B))}. \tag{8}$$

and the objective is to force Eq. (8) to be less than a predefined gain $\gamma$. Considering the squared euclidean metric as a choice for $d(.)$, one may rewrite Eq. (8) under the squared $L_2$ norm distance as

$$\begin{aligned} \frac{\boldsymbol{\epsilon}(k)^T \boldsymbol{P} \boldsymbol{\epsilon}(k)}{\|\hat{\boldsymbol{y}}(\boldsymbol{x}) - \hat{\boldsymbol{y}}(\boldsymbol{x}_B)\|^2} \leq \gamma^2, \\ \boldsymbol{\epsilon}(k)^T \boldsymbol{P} \boldsymbol{\epsilon}(k) \leq \gamma^2 \|\hat{\boldsymbol{y}}(\boldsymbol{x}) - \hat{\boldsymbol{y}}(\boldsymbol{x}_B)\|^2, \end{aligned} \tag{9}$$

which results in the final cost function as

$$J(\hat{\boldsymbol{y}}; \hat{\boldsymbol{\theta}}_B, \hat{\boldsymbol{\theta}}) = \boldsymbol{\epsilon}(k)^T \boldsymbol{P} \boldsymbol{\epsilon}(k) - \gamma^2 \|\hat{\boldsymbol{y}}(\boldsymbol{x}) - \hat{\boldsymbol{y}}(\boldsymbol{x}_B)\|^2. \tag{10}$$

**Remark 24** *Since, it is convenient and typical in classification approaches to solve the optimization problem in terms of maximum likelihood principle, one may write the learning problem as*

$$\begin{aligned} J = \quad & [-log(p(\boldsymbol{y}|\boldsymbol{x})) - log(p(\boldsymbol{y}|\boldsymbol{x}_B))] \\ - \quad & \gamma^2 d(\hat{\boldsymbol{y}}(\boldsymbol{x}), \hat{\boldsymbol{y}}(\boldsymbol{x}_B)), \end{aligned}$$

*where $p(\boldsymbol{y}|\boldsymbol{x}_B))$ & $p(\boldsymbol{y}|\boldsymbol{x})$ are parametrically estimated by $\hat{\boldsymbol{y}}$ and KL divergence can be used as a divergence measure for $d(\hat{\boldsymbol{y}}(\boldsymbol{x}), \hat{\boldsymbol{y}}(\boldsymbol{x}_B))$.*

In the current literature [27, 17, 28, 29, 15, 14, 16, 8], most of the work involves the scenario when the exact nature of $d(\hat{\boldsymbol{y}}(\boldsymbol{x}), \hat{\boldsymbol{y}}(\boldsymbol{x}_B))$ is known. This is because, samples from the target distribution are available. In contrast with [27, 17, 28, 29, 15, 14, 16, 8], we obtain samples from the neighborhood in place of these target sample.

Since we want to increase the distribution shift while reducing its impact, one may restate the overall objective in game theoretic parlance where $\hat{\boldsymbol{\theta}}_B$ aims at maximizing the DS while $\hat{\boldsymbol{\theta}}$ aims at minimizing their impact. The classifier and the distortion model are the two players in the proposed approach that are tied through the cost function. Therefore, an increase in perturbation by the distortion model increases the generalization error term in the cost function thus reducing the payoff for the classifier. Similarly, if the classifier minimizes the error of learning and generalization, the payoff for the distortion model decreases.

As a result, a strategy played by any one of the players results in an impact on the other player and vice versa. This interaction between the two players introduce the dynamics required to play the game. Mathematically, the optimization problem is designated as

$$(\boldsymbol{\theta}^*, \boldsymbol{\theta}_B^*) = \arg\min_{\hat{\boldsymbol{\theta}} \in \boldsymbol{\Omega}} \arg\max_{\hat{\boldsymbol{\theta}}_B \in \boldsymbol{\Omega}_B} J(\hat{\boldsymbol{y}}; \hat{\boldsymbol{\theta}}_B, \hat{\boldsymbol{\theta}}), \tag{11}$$

such that $(\boldsymbol{\theta}^*, \boldsymbol{\theta}_B^*)$ is the saddle point solution with $\boldsymbol{\Omega}, \boldsymbol{\Omega}_B$ representing the respective parameter spaces that are assumed to be compact.

**Remark 25** *Observe that the approximation of $E[\boldsymbol{e}_{gen}]$ depends on the training data and its neighborhood. Therefore, one is able to capture more of the data-distribution than what is represented by $\boldsymbol{\mathcal{X}}$. Thus, $\boldsymbol{\mathcal{X}}_B$ allows us to approximate the impact of heterogeneity rather than knowing it a priori in contrast with [35]. In contrast with cross-validation [36] and regularization approaches [37], the proposed approach introduces model selection into the optimization problem.*

## 4.4. DIRECT ERROR-DRIVEN LEARNING

To solve the learning problem in eq. (11), a layer-wise performance measure and the targets are defined. Let the term $\varepsilon$ is defined as $\varepsilon = \frac{dJ}{d\hat{y}(\boldsymbol{x})}$ whereas $\varepsilon_B = \frac{dJ}{d\hat{y}(\boldsymbol{x}_B)}$. The factor $\varepsilon_B$ represents the feedback from the cost function for the distortion model. Using these definitions, $H^{(i)}(\hat{\boldsymbol{y}}, \boldsymbol{x}; \hat{\boldsymbol{W}}^{(i)})$ are defined for each layer in the NN. Specifically, the layer-wise cost is defined as a function of overall error through $\boldsymbol{\delta}_k^{(i)}$ such that minimizing $H^{(i)}(\hat{\boldsymbol{y}}, \boldsymbol{x}; \hat{\boldsymbol{W}}^{(i)})$ for each $i$ minimizes the overall error. It follows that

$$H^{(i)}(\hat{\boldsymbol{W}}^{(i)}) = \frac{1}{2}\big[tr((\boldsymbol{\delta}_k^{(i)})^T \boldsymbol{Q} \hat{\boldsymbol{W}}_k^{(i)}) + \lambda \|\boldsymbol{W}_k^{(i)}\|^2\big], \tag{12}$$

where $tr(.)$ is the trace operator and $\boldsymbol{Q}$ is a positive definite symmetric matrix of choice. Note that for each layer $i$, $\boldsymbol{\delta}_k^{(i)}$ can be understood as the feedback provided by the overall cost $H(\boldsymbol{\theta}, \boldsymbol{\theta}_B)$ towards controlling the layer $(i)$. The feedback $\boldsymbol{\delta}_k^{(i)}$ is then given as $\boldsymbol{\delta}^{(i)}(k) = [f^{(i-1)}(\boldsymbol{x}) \quad f^{(i-1)}(\boldsymbol{x}_B)]\boldsymbol{\varepsilon}^T \boldsymbol{B}^{(i)}$, where $\boldsymbol{x} \in \boldsymbol{\mathcal{X}}$.

Similarly, a cost that is $H_d(\hat{\boldsymbol{\theta}}_B)$ for the distortion model is also defined based on $\epsilon_B$ such that $H_d(\hat{\boldsymbol{\theta}}_B) = \frac{1}{2}\left[tr((\boldsymbol{\delta}_d(k))^T \boldsymbol{Q}\hat{\boldsymbol{\theta}}_B(k)) - \lambda\|\hat{\boldsymbol{\theta}}_B(k)\|^2\right]$, where $\boldsymbol{\delta}^{(0)}(k)$ is defined as $\boldsymbol{\delta}_d(k) = \boldsymbol{\varepsilon}_B^T \boldsymbol{B}^{(0)}$.

Next, the feedback matrix is introduced as a variable in the layer-wise cost function and define $\boldsymbol{\mathcal{B}} = [\boldsymbol{B}^{(0)} \cdots \boldsymbol{B}^{(d)}]$. Finally, write the overall cost as the sum of layer-wise costs with three variables that is

$$H(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\theta}}_B, \boldsymbol{\mathcal{B}}) = \sum_{i=1}^{d} [\zeta^{(i)} H^{(i)}(\hat{\boldsymbol{W}}^{(i)})] - \gamma^2 H_d(\hat{\boldsymbol{\theta}}_B). \tag{13}$$

As a result, the final optimization is given as

$$(\boldsymbol{\theta}^*, \boldsymbol{\theta}_B^*, \boldsymbol{\mathcal{B}}^*) = \underset{\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\mathcal{B}}} \in \boldsymbol{\Omega}, \boldsymbol{\Omega}_{\boldsymbol{\mathcal{B}}}}{\arg\min} \; \underset{\hat{\boldsymbol{\theta}}_B \in \boldsymbol{\Omega}_B}{\arg\max} \, H(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\theta}}_B, \hat{\boldsymbol{\mathcal{B}}}), \tag{14}$$

where $\boldsymbol{\Omega}_{\boldsymbol{\mathcal{B}}}$ is the parameter space for $\boldsymbol{\mathcal{B}}$. The parameter updates at each iteration $k$ are then given as

$$\hat{\boldsymbol{\theta}}(k+1) = \hat{\boldsymbol{\theta}}(k) - \eta_1 \Delta_{\hat{\boldsymbol{\theta}}} H$$

$$\hat{\boldsymbol{\theta}}_B(k+1) = \hat{\boldsymbol{\theta}}_B(k) + \eta_2 \Delta_{\hat{\boldsymbol{\theta}}_B} H,$$

$$\boldsymbol{\mathcal{B}}(k+1) = \eta_3 \frac{\Delta_{\boldsymbol{\mathcal{B}}} H}{\|\boldsymbol{\mathcal{B}}(k)\|}$$

where $\eta_1$, $\eta_2$ and $\eta_3$ are the respective learning rates.

**Remark 26** *Since, the cost function consists of an $L_2$ norm penalty, the updates for $\hat{\boldsymbol{\theta}}_B$ will have a decaying term. The decaying term will then ensure that $\|\hat{\boldsymbol{\theta}}_B\|$ is bounded. As a consequence, $g(.,\hat{\boldsymbol{\theta}}_B)$ is bounded and Lemma. 6 holds. Furthermore, one may observe that the weights of the neural networks are bounded too.*

One may observe that the proposed formulation represents a mixed strategy two player game. Therefore, a saddle point solution is not guaranteed and it must be shown [38]. Furthermore, within the proposed formulation, one may observe that the a cost function is derived for each layer in the NN, which makes the existence of a saddle point non-trivial. To this end, we would first show that $H$ satisfies the conditions under which a saddle point solution is possible. We demonstrate that the conditions are true even when the number of layers in the neural network as well as the number of neurons at every layer tend to infinity. Next, under the strategies chosen by the two players, it will be shown in Theorem 3 that proposed formulation would achieve Nash equilibrium. With a little abuse of notation let $\boldsymbol{\theta}^*$ and $\hat{\boldsymbol{\theta}}$ to include $\boldsymbol{\mathcal{B}}^*$ and $\boldsymbol{\mathcal{B}}$ respectively.

**Theorem 3** *Let $H$ be defined as shown in Eq.* (13) *and $Q$ be a positive definite matrix. Further, let the solution pair be given as $(\boldsymbol{\theta}_B^*, \boldsymbol{\theta}^*)$ and let the parameter set be given as $(\Omega_{\boldsymbol{\theta}_B^*}, \Omega)$. Consider the case when the number of layers as well as the number of neurons in the NN tend to infinity and let $\sum_{i=1}^{d} \zeta^i = L, \sum_{j=1}^{\eta^{(i)}} \nu_j = L$, where $L \in \mathbb{R}$. Furthermore, consider that $\gamma > 0$ and it follows that $H$ is a linear combination of quadratic forms that are convex with respect to the parameters of the neural network. Similarly, $H$ is a linear combination of quadratic forms that are concave with respect to the parameters of the distortion model. Under the assumption that the parameters sets are compact, it can be concluded that the two player zero-sum game represented by $H$ has a saddle point denoted by $(\boldsymbol{\theta}_B^*, \boldsymbol{\theta}^*)$. Proof: Please see the appendix.*

**Remark 27** *The conditions $\eta^{(i)} \to \infty$ implies the big data case when the number of dimensions in the data are very large. This is due to the fact that the large dimensions in the data typically requires a larger NN for optimality. As a result, the theorem above is valid for a very deep and large neural network.*

**Remark 28** *The conditions $\sum_{i=1}^{d} \zeta^i = L, \sum_{j=1}^{\eta^{(i)}} \nu_j = L$, are important to ensure that the cost function value is not numerically unbounded. Further $\zeta^i$ signifies the contribution of each layer in the NN to the cost function and $\nu_j$ signifies the contribution of each neuron at every* *layer in the neural network. In practical scenarios, the number of layers and the number of neurons in the NN can be adjusted to make sure that the contributions of different neurons and different layers in the NN is not very small.*

**Remark 29** *The choice of $\gamma$ is important to enable the solution space. A large value of $\gamma$ would increase importance of distortion model in the learning and the neural network would focus more on the the domain shifted data-distribution. On the other hand, a small value of $\gamma$ would better the accuracy but would compensate less for the heterogeneity in the data-distribution.*

Finally, the proposed methodology is applied on several datasets and the results are presented in the following section. An algorithm for the overall methodology is presented in Algorithm 6. Using this algorithm, the results of this methodology are detailed next.

**Algorithm 6** Two step learning algorithm for DNN

```
1: Input: $\mathcal{X}$,
2: for $k = 1 \rightarrow \Gamma$, where $\Gamma$ is the number of epochs do
3:     for each batch in $\mathcal{X}$ do
4:         Generate alternate batch
5:         Learning Step
               $\hat{\boldsymbol{\theta}}(k+1) = \hat{\boldsymbol{\theta}}(k) - \eta_1 \Delta_{\hat{\boldsymbol{\theta}}} H$
               $\hat{\boldsymbol{\theta}}_B(k+1) = \hat{\boldsymbol{\theta}}_B(k) + \eta_2 \Delta_{\hat{\boldsymbol{\theta}}_B} H,$
               $\mathcal{B}(k+1) = \eta_3 \frac{\Delta_{\mathcal{B}} H}{||\mathcal{B}(k)||}$
6:     end for
7: end for
```

Table 2. Parameters of the different models used in this paper.

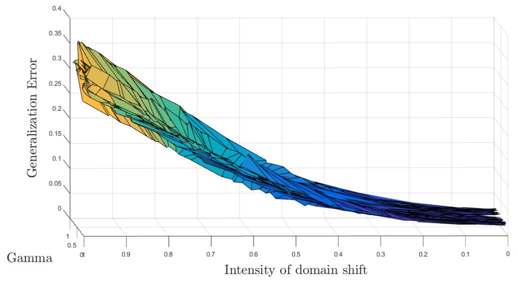
| Architecture | Parameters |
|---|---|
| CNN | $n_{conv} = 2$, $n_{layers} = 2$,<br>filter size = 5, pool shape = (3,3),<br>hln= 128, $\alpha = 0.01$, $b = 100$, $\lambda = 0.001$<br>$n_{neigh} = 2$, $\rho = 1$ |
| MLNN | $d = 7$, hln= 128, $\alpha = 0.01$, $b = 100$, $\lambda = 0.001$<br>$n_{neigh} = 2$, $\rho = 1$ |
| SAE | $n_{units} = 3$,<br>hln= 128, $\alpha = 0.01$, $b = 100$, $\lambda = 0.001$<br>$n_{neigh} = 2$, $\rho = 1$ |
| DAE | $n_{units} = 3$,<br>hln= 128, $\alpha = 0.01$, $b = 100$, $\lambda = 0.001$<br>$n_{neigh} = 2$, $\rho = 1$ |

## 5. RESULTS AND DISCUSSION

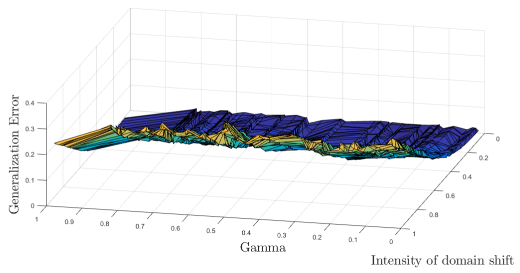
A total of ten data-sets are used for analysis and the statistics for these data-sets are summarized in Table 4. In all of the data-sets considered here, 80% of the available data is chosen for learning and twenty percent is used to test the trained model [30]. All of the results demonstrated in the next section are calculated from the data used to test the trained model. The splits are constructed randomly.

All of the results in this section are averaged over 100 initial conditions. The software package Tensor-flow with Python is used for all of the experiments in this paper. The hyper-parameters are summarized in Tables 4 and 2. In Tables 2 and 4, hln refers to the number of hidden layer neurons at each layer; b denotes the batch size and $n_{neigh}$ represents the number of data-points in the neighborhood corresponding to each data-point in the data-set. Also, $n_{conv}$ represents the number of convolutional layer and $n_{layers}$ represents the number of feed-forward layers. Affine transformation is chosen for the distortion model.

We start by analyzing the MNIST data-set. First, we show that the use of distortion model improves performance, when DS is introduced in the data-set.



(a)



(b)

Figure 3. (a) Surface plot for change in $\gamma$; (b)Surface plot for change in intensity shift.

## 5.1. TOY PROBLEM - MNIST DATA-SET

To simulate the notion of DS and test the ability of the methodology to compensate for it, Gaussian-distributed and uniformly-distributed perturbations are introduced into the MNIST data-set. In other words, additional data-points are added into the test sample by applying an affine transformation to each data-point in the test sample when the parameters of the affine transformation are sampled using uniform and Gaussian distribution. To simulate the extent of the DS, we vary the norm of the affine transformation. The objective is to study the impact of these distortions on learning effectiveness that is the testing accuracy.

For sampling the distortions in the test phase, the mean vector for the Gaussian distribution is chosen as a zero, whereas the variance matrix for the perturbations is chosen as $\rho \boldsymbol{I}$, with $\boldsymbol{I}$ being an appropriate identity matrix and $\rho$ being the variance of choice. The parameters for the uniform-distribution for the affine transformation are chosen as $[-\rho, \rho]$. Five layer NNs with relu activation functions are utilized with the learning rates chosen as

Table 3. Mean test accuracies for the proposed framework compared with traditional SGD for the MNIST data-set.

| | **SGD** | [11] | **Minimax** |
|---|---|---|---|
| relu (Gaussian) | 0.83 | 0.91 | 0.97 |
| tanh (Gaussian) | 0.82 | 0.87 | 0.97 |

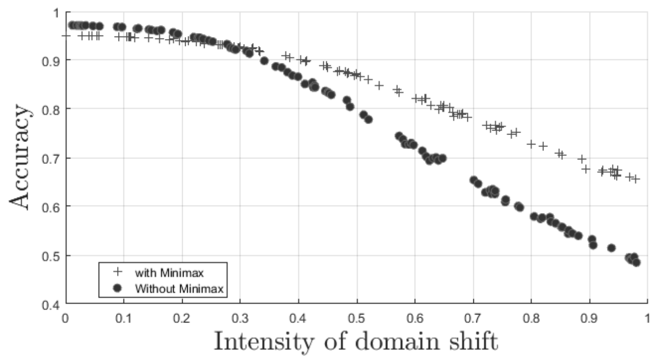
$\eta_1 = 0.001$ and $\eta_2 = 0.1$. In Table 3, the results of a total of 100 runs are recorded. The methodology in [11] is implemented with $\rho = 1$ with both uniform and Gaussian mean shift.

Observe the performance improvement for the minimax framework relative to the case when the neighborhood is defined directly. Furthermore, observe the 6 % improvement in the performance between [11] and the minimax framework in the case when uniformly distributed complications are present in the data.

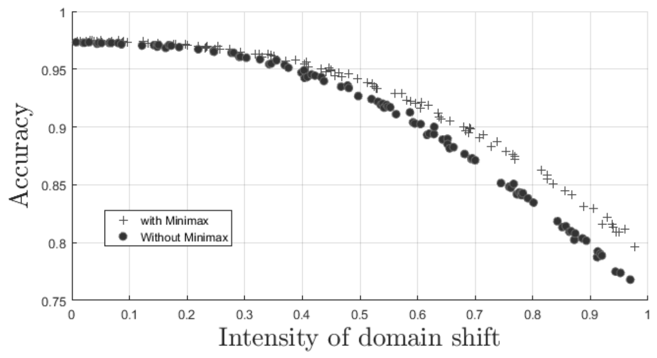
On the other hand, one may observe that when the distortions are sampled from Gaussian distribution, we see a 10 % improvement over [11]. Furthermore note that, Gaussian perturbation in the data effects both the first and second order moments of the data-distribution, the compensation for which is not possible with [11] as observed from Table 3 where the minimax framework outperforms both SGD and [11] in the presence of Gaussian distortions. Moreover, note that the key difference between [11] and the current work is that, the neighborhood estimation is part of the optimization here.

In Figures. 4b and 4a, the performance of the classifier for increasing intensity of the DS is demonstrated. One may observe that the conclusion form Table 3 carries over to these results. The proposed framework is therefore resilient to an increase in the DS.

Next, we evaluate the methodology when the amount of noise in the data increases and the results are shown in Figure 5b. The generalization performance of the proposed framework is studied for increasing dissimilarity of the test data introduced using synthetic



(a)

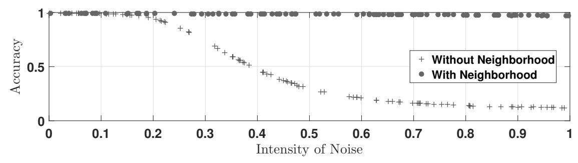


(b)

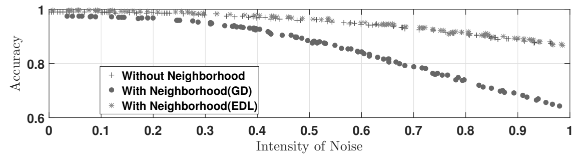
Figure 4. (a) Accuracy with respect to an increase in norm intensity for MNIST data-set with respect to Gaussian distributed changes in the data.; (b) Accuracy with respect to an increase in norm intensity for MNIST data-set with respect to Uniformly distributed changes in the data.

Table 4. % accuracies for the various data-sets with the proposed framework and four different learning algorithms with generalization error in the parenthesis.

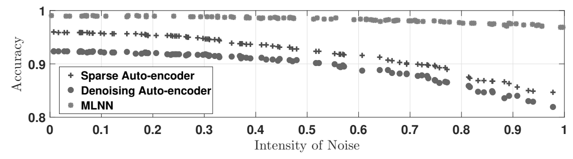
| Datasets | $p$ | $n$ | $\mathcal{F}$ | Model Parameters | **DFA** | **FA** | **SGD** | **EDL** |
|---|---|---|---|---|---|---|---|---|
| **Rolling** [39] | 11 | 35000 | 4 | $\alpha = 0.01, b = 100, d = 7, hln = 100,$ $\lambda = 0.001, n_{neigh} = 2, \rho = 1$ | 99(0.0001) | 99(0.0001) | 99(0.6) | 99(0.0) |
| **Sensorless** [40] | 48 | 78000 | 11 | $\alpha = 0.01, b = 100, d = 8, hln = 100,$ $\lambda = 0.001, n_{neigh} = 2, \rho = 1$ | 94(0.008) | 94(0.008) | 95(0.008) | 94(0.002) |
| **MNIST** [41] | 784 | 72000 | 10 | $\alpha = 0.01, b = 100, d = 8, hln = 100,$ $\lambda = 0.001, n_{neigh} = 2, \rho = 1$ | 92(0.1) | 94(0.2) | 95(1) | 97(0.6) |
| **NotMnist** [42] | 784 | 81000 | 10 | $\alpha = 0.01, b = 100, d = 9, hln = 100$ $\lambda = 0.001, n_{neigh} = 2, \rho = 1$ | 90(1) | 82(1.8) | 88(2) | 92(0.8) |
| **CIFAR10** [43] | 3072 | 50000 | 10 | $\alpha = 0.01, b = 100, d = 10, hln = 100$ $\lambda = 0.001, n_{neigh} = 2, \rho = 1$ | 81(7.21) | 80(8.00) | 82(12.11) | 84(8.77) |
| **Arcene** [44] | 10000 | 100 | 2 | $\alpha = 0.01, b = 100, d = 9, hln = 100, \lambda = 0.001$ $\lambda = 0.001, n_{neigh} = 2, \rho = 1$ | 69(0) | 61(0) | 51(0) | 78(0) |
| **Dexter** [44] | 20000 | 300 | 2 | $\alpha = 0.01, b = 100, d = 8, hln = 100$ $\lambda = 0.001, n_{neigh} = 2, \rho = 1$ | 71(0.7) | 77(0.9) | 81(1.1) | 80(1) |
| **Gisette** [44] | 5000 | 6000 | 2 | $\alpha = 0.01, b = 100, d = 8, hln = 100,$ $\lambda = 0.001, n_{neigh} = 2, \rho = 1$ | 92(0.25) | 93(0.21) | 91(0.3) | 98(0.07) |



(a) CNN



(b) MLNN



(c) SAE and DAE

Figure 5. Change in the size of the neighborhood versus (top) accuracy (bottom) generalization error. Results presented in these plots are averaged over two types of perturbations that are Gaussian and Uniform distributed.

Table 5. Average accuracies.

| Data-set | $n$ | $p$ | $\mathcal{F}$ | Accuracy |
|---|---|---|---|---|
| Synthetic | 20000 | 2 million | 10 | 94.10 |
| Image-Net[45] | 784 | 12614060 | 10 | 56.21 |

Table 6. Mean test accuracies with the standard error in accuracy for the MNIST data-set.

| | **SGD** | **PF+GD** |
|---|---|---|
| relu (Gaussian) | 0.83(0.07) | 0.91(0.093) |
| relu (Uniform) | 0.85(0.10) | 0.93(0.018) |
| tanh (Gaussian) | 0.82(0.006) | 0.87(0.003) |
| tanh (Uniform) | 0.86(0.10) | 0.89(0.018) |

perturbations. The neighborhood size for training is kept as one and the average accuracies and generalization error over one hundred initial conditions of weights are illustrated in Figure 5b. The deterioration in performance for the proposed framework with an increase in the perturbations is smaller with the proposed framework. Both learning regimes, EDL and GD, show improved performance. In Figure 5b, SGD results in accuracy that is a little higher than that achieved by EDL when no dissimilarity is present. However, EDL is very resilient to increasing dissimilarity in the data. In other words, the proposed approach, while exhibiting comparable best case accuracy, indicates improved resilience in the presence of data-noise and heterogeneity.

Next, we test the performance of convolutional neural networks (CNN) [46] and multilayer neural networks (MLNN), with respect to the proposed methodology. The hyper-parameters corresponding to these architectures are summarized in Table 2. For both CNN and MLNN, it is consistently observed that a significant improvement in performance is achieved when uniformly and Gaussian distributed perturbations are introduced to construct the neighborhood.

Next, the performance of the proposed methodology is compared with sparse auto-encoders [47] and denoising auto-encoders [48]. The hyper-parameters are again summarized in Table 2. The results are demonstrated in Figure 5c. The best case accuracies for varying amount of noise in the data-set is achieved by the proposed approach. The proposed methodology is built to address scenarios where the test data-set is different from the training data-set, whereas sparse auto-encoders and denoising auto-encoders are constructed to look for and remove noisy patterns in the training data-set. Moreover, explicit modification of the objective function appears to help and thus improve performance within the proposed approach.

Next, the performance of the proposed framework is studied relative to the regular SGD for the MNIST data-set, the results are summarized in Table 6. The accuracies are higher for relu compared to tanh activation functions. The improved performance of the relu activation function over tanh is well recorded in the literature [49]. The standard error of accuracy for the proposed framework is much smaller compared to traditional SGD. Therefore, learning with the proposed framework is resilient towards varied initial conditions and the results are statistically significant. The impact of the values of gamma on learning within the proposed framework is illustrated next.

The extra penalty term in the cost function controls the size of the neighborhood by determining the amount of perturbations, the distortion model may introduce in the data-set. As a result, an appropriate choice of $\gamma$, the trade off between the neighborhood size and the accuracy can be addressed and the size of the neighborhood can be controlled. This impact of $\gamma$ on the learning performance is discussed next.

## 5.2. IMPACT OF $\gamma$

A large influence on the neighborhood and the effect on the generalization error is due to the use of the term $\gamma$ in the optimization problem. In Figure 3a, the behavior of the generalization error with change in the intensity of DS is illustrated.

One may observe by a drop in generalization error when the value of gamma is close to one that with an increase in the value of gamma, the intensity of DS has less impact on the learning efficiency. The surface plot also indicates the changes in the generalization error with change in intensity. It may be observed that the values of generalization errors are large when the intensity of DS is large and vice versa which is observed in Figure 3b. Next, the performance of the proposed method is tested on classification.

## 5.3. CLASSIFICATION

Four learning paradigms, namely DFA (Direct Feedback Alignment) [50], SGD (Stochastic Gradient Descent) [10], FA (Feedback Alignment) [51] and EDL (Error-driven Learning), are tested with all the data-sets listed in Table 4. The hyper-parameters for different models used in the paper are summarized in Table 4.

Consistent performance of the proposed framework with EDL is observed across data-sets as is seen in Table 4. Reasonable accuracies are observed for the large dimensional data-sets and others provide marginal improvement for typical SGD. For the dexter data-set, our EDL method achieved similar accuracies compared to standard SGD but indicates significant improvement over DFA and FA. The guided choice of the feedback matrix appears to assist EDL.

Next, the generalization performance of the methodology for different data-sets is presented in Table 4. It is observed that the lowest generalization error is achieved for the proposed framework with the EDL learning scheme. Even for the case of gradient-based updates on the proposed framework, there is a significant reduction in the generalization error over the traditional SGD. Improved generalization errors indicate that the impact of challenges such as heterogeneity and data-noise is indeed mitigated with the proposed framework.

## 5.4. BIG DATA APPLICATION

Big data is commonly defined as the scenario where the analysis of a big data-set cannot be conducted on a single core CPU. Therefore, to demonstrate the use of proposed methodology in such an environment, our approach is implemented on a multi-core CPU. The results are demonstrated on two large data-sets where the first is the popular ImageNet data-set [45], and the second is the synthetic data-set.

We construct a ten class problem from the ImageNet data-set by sampling with replacement. In the end, the data-set is about 138 GB in size consisting of ten classes. Similarly, the synthetic data-set is composed of 20,000 features and 2 million data-points. While generating this data-set, clusters of normally distributed points are created about vertices of an 20,000-dimensional hypercube with sides of a pre-defined length. Both of these data-sets are extremely large. The results, as summarized in Table 5, indicate the ability of the methodology to obtain reasonable performance.

The results described in this section suggest that the proposed min-max framework with EDL may serve as a promising prospect on the problem of classification, especially in a large dimensional scenario with much noise, as can be observed in the results.

## 6. CONCLUSIONS

A systematic approach was presented to design a classifier in the presence of challenges such as heterogeneity, vanishing gradients and noise. The two player game setup provides the optimal compensation for the classifier and the corresponding distortion. Due to this the proposed approach is more resilient to distribution changes that is the domain shift.

The direct use of the overall error signal in the learning process mitigates the vanishing gradient issue. The proposed framework combined with EDL appears to outperform other approaches such as DFA and FA in all of the data-sets under analysis. The main drawback of the proposed approach is the need to select an appropriate distortion model.

Overall, the proposed approach appears to provide a 5% improvement in generalization error and a 10% improvement in accuracy over SGD in the presence of domain shift. Future effort will address the challenges such as non-stationary distributions and incidental endogeneity.

## ACKNOWLEDGMENT

This research was supported in part by NSF I/UCRC award IIP 1134721 and Intelligent Systems Center.

## APPENDIX

**6.0.0.1. Proof:.** [Lemma 1 Proof] Given a random variable $\boldsymbol{X}$, let the moment generating function $\Psi(t)$ [52] be defined as

$$\Psi(t) = E\{\exp^{t\boldsymbol{X}}\}, \forall t \in \mathbb{R}, \tag{15}$$

where $\boldsymbol{X}$ is the random variable and $E$ is the expectation. Consider the expansion of the $\Psi(t)$ using the series expansion of exponential functions and apply the transformation $g(.)$ [52] to get

$$\Psi(t) = \sum_{i=0}^{\infty} \frac{t^i}{i!} E[(g(\boldsymbol{X}))^i]. \tag{16}$$

Under the assumption that the transformation is bounded such that $|g(\boldsymbol{X})| \leq g^*(\boldsymbol{X}), \forall i$ we get by Lebesgue dominated convergence theorem that

$$\begin{aligned} \Psi(t) &\leq \sum_{i=0}^{\infty} \frac{t^i}{i!} E[g^*(\boldsymbol{X})], \\ &\leq 1 + tE[g(\boldsymbol{X})] + \frac{t^2}{2!} E[g(\boldsymbol{X})]^2 + \cdots . \end{aligned} \tag{17}$$

where the first raw moment under first order Taylor series approximation is given as

$$\begin{aligned} \mu_1 &= E[g(\boldsymbol{X})], \\ &\leq E[g(E[\boldsymbol{X}]) + g'(E[\boldsymbol{X}])(\boldsymbol{X} - E[\boldsymbol{X}])]. \end{aligned}$$

Similarly, the second order raw moment can be achieved using the Jenson's inequality resulting in

$$\mu_2 = E[g(\boldsymbol{X}^2)],$$
$$\leq E[g(E[\boldsymbol{X}^2]) + g'(E[\boldsymbol{X}^2])(X^2 - E[\boldsymbol{X}^2])].$$

The question of interest is to see what happens to these moments when the number of data-points in the population tend to infinity under specific choices of transformation. Two cases of these transformations are considered and the corresponding moments are derived below.

*Case 1:* In a simple case, consider affine transformation $\boldsymbol{\mathcal{X}}$ and consider $\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}}$ to be the parameters of the affine transformation such that

$$vec(\hat{\boldsymbol{A}}) \sim p_A(\boldsymbol{\mu}_{\hat{A}}, \boldsymbol{\mathcal{S}}_{\hat{A}}), vec(\hat{\boldsymbol{b}}) \sim p_b(\boldsymbol{\mu}_{\hat{\boldsymbol{b}}}, \boldsymbol{\mathcal{S}}_{\hat{\boldsymbol{b}}}),$$

where $p_.(.)$ is a probability distribution of choice characterized by the first two moments: mean $\boldsymbol{\mu}_{(.)}$ and covariance matrix $\boldsymbol{\mathcal{S}}_{(.)}$. Note that $vec()$ denotes vectorization of a matrix. Thus $g(\boldsymbol{x}; \hat{\boldsymbol{\theta}}_B) = \hat{\boldsymbol{A}}^T \boldsymbol{x} + \hat{\boldsymbol{b}}$ and it follows that

$$\mu_1 \leq E[g(E[\boldsymbol{X}]) + g'(E[\boldsymbol{X}])(\boldsymbol{X} - E[\boldsymbol{X}])].$$
$$\leq E[\hat{\boldsymbol{A}}^T E[\boldsymbol{X}] + \hat{\boldsymbol{b}}] + \hat{\boldsymbol{A}}^T(\boldsymbol{X} - E[\boldsymbol{X}])].$$
$$\leq \hat{\boldsymbol{A}}^T E[\boldsymbol{X}] + \hat{\boldsymbol{b}},$$

which implies that

$$\mu_1 \leq \hat{\boldsymbol{A}}^T E[\boldsymbol{X}] + \hat{\boldsymbol{b}} \tag{18}$$

Then, we can write using Cauchy's inequality

$$\|\mu_1\| \leq \|\hat{\boldsymbol{A}}\| \|E[\boldsymbol{X}]\| + \|\hat{\boldsymbol{b}}\|. \tag{19}$$

Simplification with limit $n, p \longrightarrow \infty$ through the law of large numbers reveals that $E[\boldsymbol{X}]$ will converge to the population estimate for the random variable $\boldsymbol{X}$. Let the population estimate be written as $\mu(\boldsymbol{X})$ and the first moment converges to

$$||\mu_1|| \leq \lambda_{max}(\hat{\boldsymbol{A}})||\mu(\boldsymbol{X})|| + ||\hat{\boldsymbol{b}}||, \tag{20}$$

Similarly, the second order raw moment can be achieved as

$$\begin{aligned} \mu_2 &\leq E[g(\boldsymbol{X})^2] + 2E[g(E[\boldsymbol{X}])g'(E[\boldsymbol{X}])(X - E[\boldsymbol{X}]]. \\ &\leq E[\boldsymbol{X}^T\hat{\boldsymbol{A}}\hat{\boldsymbol{A}}^T\boldsymbol{X}] + 2\hat{\boldsymbol{A}}^T E[\boldsymbol{X}]\hat{\boldsymbol{b}} + E[\hat{\boldsymbol{b}}^T\hat{\boldsymbol{b}}]. \end{aligned}$$

Simplification with limit $n, p \longrightarrow \infty$ reveals

$$\begin{aligned} ||\mu_2|| \leq \lim_{n,p\to\infty} \big[&Tr(\hat{\mathcal{A}}\boldsymbol{\Sigma}(\boldsymbol{X})) \\ &+||\boldsymbol{\mu}(\boldsymbol{X})^T\boldsymbol{\Sigma}(\boldsymbol{X})\boldsymbol{\mu}(\boldsymbol{X})|| \\ &+||2\hat{\boldsymbol{A}}^T\boldsymbol{\mu}(\boldsymbol{X})\boldsymbol{b} + E[\hat{\boldsymbol{b}}^T\hat{\boldsymbol{b}}]||\big], \end{aligned} \tag{21}$$

where $\boldsymbol{\Sigma}(\boldsymbol{X}) = \boldsymbol{X}^T\boldsymbol{X}$ is the asymptotic covariance matrix and $\hat{\mathcal{A}} = \hat{\boldsymbol{A}}\hat{\boldsymbol{A}}^T$. Finally, we get

$$\begin{aligned} ||\mu_2|| &\leq \lambda_{max}(\hat{\mathcal{A}})\lambda_{max}(\boldsymbol{\Sigma}(\boldsymbol{X})) + ||\hat{\boldsymbol{b}}|| \\ &+[\lambda_{max}(\boldsymbol{\Sigma}(\boldsymbol{X})) + 2\lambda_{max}(\hat{\boldsymbol{A}})||\hat{\boldsymbol{b}}||]||\boldsymbol{\mu}(\boldsymbol{X})|| \end{aligned}$$

*Case 2:* In a general case, let $g(\boldsymbol{X})$ be a generic nonlinear transformation such that $g(\boldsymbol{X}) = \hat{\boldsymbol{\theta}}_B^T\Phi(\boldsymbol{X})$. Following the same steps as before, we can conclude the first moment as

$$||\mu_1|| \leq ||\hat{\boldsymbol{\theta}}_B||||\mu(\Phi(\boldsymbol{X}_j))||. \tag{22}$$

The second moment is then given as

$$\|\mu_2\| \leq \lim_{n,p\to\infty} \|E[\Phi(\boldsymbol{X})^T\hat{\boldsymbol{\theta}}_B\hat{\boldsymbol{\theta}}_B^T\Phi(\boldsymbol{X})]\|, \leq \lim_{n,p\to\infty} Tr(\boldsymbol{\mathcal{A}}\Sigma(\Phi(\boldsymbol{X}))). \tag{23}$$

The asymptotic result can finally be achieved as

$$\|\mu_2\| \leq \quad \lambda_{max}(\boldsymbol{\mathcal{A}})\lambda_{max}(\Sigma(\Phi(\boldsymbol{X})),. \tag{24}$$

$\square$

**6.0.0.2. Proof:.** [Theorem 1 proof] Observe that

$$H^{(i)} = \big[tr((\boldsymbol{\delta}_k^{(i)})^T\boldsymbol{Q}^{(i)}\hat{\boldsymbol{W}}_k^{(i)}) = tr((\boldsymbol{Q}^{(i)}\hat{\boldsymbol{W}}_k^{(i)}(\boldsymbol{\delta}_k^{(i)})^T) = tr(\boldsymbol{Q}^{(i)}\boldsymbol{X}^{(i)}) \tag{25}$$

where $\boldsymbol{X}^{(i)} = \hat{\boldsymbol{W}}_k^{(i)}(\boldsymbol{\delta}_k^{(i)})^T$ is a square matrix and $\boldsymbol{Q}^{(i)}$ is symmetric and positive definite matrix. Rewrite $\boldsymbol{Q}^{(i)} = \sum_{j=1}^{\eta^{(i)}} \nu_j \boldsymbol{p}_j^T\boldsymbol{p}_j$, where $\eta^{(i)}$ is the number of hidden layer nodes at layer $i$. Substitution into Eq. (25) and simplification reveals

$$H^{(i)} = tr(\boldsymbol{Q}^{(i)}\boldsymbol{X}^{(i)}) = \sum_{j=1}^{\eta^{(i)}} \nu_j(tr(\boldsymbol{p}_j\boldsymbol{X}\boldsymbol{p}_j^T))),$$

which is a linear combination of non-negative values [53]. Consider the case when both $d$ and $\eta^{(i)}$ tend to infinity and the formulation can be alternatively interpreted as

$$\begin{aligned}\boldsymbol{\mathcal{H}} &= \lim_{\eta^{(i)},d\to\infty} \sum_{i=1}^{d} \zeta^i H^{(i)} = \lim_{\eta^{(i)},d\to\infty} \sum_{i=1}^{d} \zeta^i tr(\boldsymbol{Q}^{(i)}\boldsymbol{X}^{(i)}) \\ &= \lim_{\eta^{(i)},d\to\infty} \sum_{i=1}^{d} \zeta^i \sum_{j=1}^{\eta^{(i)}} \nu_j(tr(\boldsymbol{p}_j\boldsymbol{X}\boldsymbol{p}_j^T))).\end{aligned} \tag{26}$$

Let $\sum_{i=1}^{d} \zeta^i = L$ and $\sum_{j=1}^{\eta^{(i)}} \nu_j = L$, it follows that the term $\boldsymbol{\mathcal{H}}$ is a bounded sum provided $tr(\boldsymbol{p}_j \boldsymbol{X} \boldsymbol{p}_j^T)))$ is bounded. However, this depends on the value of the weights and $\delta^{(i)}$. The weights are bounded because of the choice of the update rule (a first order difference equation results in a bounded sequence). Furthermore, $\delta^{(i)}$ is bounded because the cost function is Lipschitz. With these observations, it follows that $\sum_{i=1}^{d} H^{(i)}$ for all the layers in the neural networks are a linear combination of standard quadratic forms that are independently convex [53] with respect to the parameters of the neural network. Next, it can be observed from $H^{(0)}(\hat{\boldsymbol{\theta}}_B)$

$$\begin{aligned} H^{(0)} &= -\gamma^2 tr((\boldsymbol{\delta}_k^{(0)})^T \boldsymbol{Q}^{(0)}(\hat{\boldsymbol{\theta}}_B)_k) \\ &= -\gamma^2 tr((\boldsymbol{Q}^{(0)}(\hat{\boldsymbol{\theta}}_B)_k \boldsymbol{\delta}_k^{(0)})^T) \\ &= -\gamma^2 tr(\boldsymbol{Q}^{(0)} \boldsymbol{X}^{(0)}) \end{aligned}$$

where $\boldsymbol{X}^{(0)} = ((\hat{\boldsymbol{\theta}}_B)_k \boldsymbol{\delta}_k^{(0)})^T$ is a square matrix and $\boldsymbol{Q}^{(0)}$ is symmetric and positive definite matrix. It follows that following the principles from earlier, $H^{(0)}$ is a concave function provided $\gamma > 0$.

Note that the overall cost function is $H^{(0)} + \sum_{i=1}^{d} H^{(i)}$. Moreover, $\sum_{i=1}^{d} H^{(i)}$ is only dependent on the parameters of the neural network and $H^{(0)}$ depends purely on the parameters of the distortion model. The second derivative of the overall cost function with respect to $\hat{\boldsymbol{\theta}}$ reveals a convex cost function because the term due to the distortion model would result in a zero. Similarly, the second derivative of the overall cost function with respect to $\hat{\boldsymbol{\theta}}_B$ reveals a concave cost function. It follows that the overall cost function is convex relative to the parameters of the neural network while it is concave with respect to the parameters of the distortion model. As a consequence, $\boldsymbol{\theta}_B^*$ is a maximizer for the cost function [54]. Similarly, it follows that $\boldsymbol{\theta}^*$ is a minimizer for the cost function due to convexity with respect to $\hat{\boldsymbol{\theta}}$. Thus, invoking the minimax theorem in [55] it can be seen that $(\boldsymbol{\theta}_B^*, \boldsymbol{\theta}^*)$ is a saddle point solution to the game and the game admits a Nash equilibrium. $\square$

# REFERENCES

[1] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, “Learning in nonstationary environments: A survey,” *IEEE Computational Intelligence Magazine*, vol. 10, no. 4, pp. 12–25, 2015.

[2] S. J. Pan, Q. Yang *et al.*, “A survey on transfer learning,” *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[3] J. Fan, F. Han, and H. Liu, “Challenges of big data analysis,” *National science review*, vol. 1, no. 2, pp. 293–314, 2014.

[4] T. Xiao, H. Li, W. Ouyang, and X. Wang, “Learning deep feature representations with domain guided dropout for person re-identification,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1249–1258.

[5] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting.” *Journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[6] J. Lemley, S. Bazrafkan, and P. Corcoran, “Smart augmentation learning an optimal data augmentation strategy,” *IEEE Access*, vol. 5, pp. 5858–5869, 2017.

[7] Y. Taigman, A. Polyak, and L. Wolf, “Unsupervised cross-domain image generation,” *arXiv preprint arXiv:1611.02200*, 2016.

[8] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, “Domain-adversarial training of neural networks,” *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[9] T. Basar and P. Bernhard, *H-infinity optimal control and related minimax design problems: a dynamic game approach.* Springer Science & Business Media, 2008.

[10] M. Hardt, “Train Faster, Generalize Better: Stability of Stochastic Gradient Descent,” *Mathematical and Computational Foundations of Learning Theory*, p. 64, 2015.

[11] R. Krishnan, S. Jagannathan, and V. Samaranayake, “Direct error driven learning for deep neural networks with applications to big data,” *Procedia Computer Science*, vol. 144, pp. 89–95, 2018.

[12] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” *arXiv preprint arXiv:1412.6572*, 2014.

[13] S. Sonoda and N. Murata, “Neural network with unbounded activation functions is universal approximator,” *Applied and Computational Harmonic Analysis*, 5 2015.

[14] M. Long, J. Wang, Y. Cao, J. Sun, and S. Y. Philip, “Deep learning of transferable representation for scalable domain adaptation,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 8, pp. 2027–2040, 2016.

[15] M. Baktashmotlagh, M. T. Harandi, B. C. Lovell, and M. Salzmann, “Unsupervised domain adaptation by domain invariant projection,” in *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE, 2013, pp. 769–776.

[16] L. Wen, L. Gao, and X. Li, “A new deep transfer learning based on sparse auto-encoder for fault diagnosis,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2017.

[17] M. Chen, Z. Xu, K. Weinberger, and F. Sha, “Marginalized denoising autoencoders for domain adaptation,” *arXiv preprint arXiv:1206.4683*, 2012.

[18] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan, “Domain separation networks,” in *Advances in Neural Information Processing Systems*, 2016, pp. 343–351.

[19] A. Rozantsev, M. Salzmann, and P. Fua, “Beyond sharing weights for deep domain adaptation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[20] P. Haeusser, T. Frerix, A. Mordvintsev, and D. Cremers, “Associative domain adaptation,” in *International Conference on Computer Vision (ICCV)*, vol. 2, no. 5, 2017, p. 6.

[21] K. Saito, K. Watanabe, Y. Ushiku, and T. Harada, “Maximum classifier discrepancy for unsupervised domain adaptation,” *arXiv preprint arXiv:1712.02560*, vol. 3, 2017.

[22] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, “Adversarial discriminative domain adaptation,” in *Computer Vision and Pattern Recognition (CVPR)*, vol. 1, no. 2, 2017, p. 4.

[23] M.-Y. Liu and O. Tuzel, “Coupled generative adversarial networks,” in *Advances in neural information processing systems*, 2016, pp. 469–477.

[24] D. Mishkin and J. Matas, “All you need is a good init,” *arXiv preprint arXiv:1511.06422*, 2015.

[25] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” *arXiv preprint arXiv:1312.6199*, 2013.

[26] K. Saito, Y. Ushiku, and T. Harada, “Asymmetric tri-training for unsupervised domain adaptation,” *arXiv preprint arXiv:1702.08400*, 2017.

[27] J. Blitzer, S. Kakade, and D. Foster, “Domain adaptation with coupled subspaces,” in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 173–181.

[28] B. Gong, Y. Shi, F. Sha, and K. Grauman, “Geodesic flow kernel for unsupervised domain adaptation,” in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2066–2073.

[29] Y. Wang, W. Li, D. Dai, and L. Van Gool, “Deep domain adaptation by geodesic distance minimization,” *arXiv preprint arXiv:1707.09842*, 2017.

[30] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[31] P. Niyogi and F. Girosi, “On the relationship between generalization error, hypothesis complexity, and sample complexity for radial basis functions,” *Neural Computation*, vol. 8, no. 4, pp. 819–842, 1996.

[32] D. M. Witten, R. Tibshirani, and T. Hastie, “A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis,” *Biostatistics*, vol. 10, no. 3, pp. 515–534, 2009.

[33] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[34] R. Elwell and R. Polikar, “Incremental learning of concept drift in nonstationary environments,” *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1517–1531, 2011.

[35] J. Pearl, “Detecting latent heterogeneity,” *Sociological Methods & Research*, vol. 46, no. 3, pp. 370–389, 2017.

[36] N. Xu, J. Hong, and T. C. Fisher, “Generalization error minimization: a new approach to model evaluation and selection with an application to penalized regression,” *arXiv preprint arXiv:1610.05448*, 2016.

[37] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[38] R. Isaacs, “Differential games ii,” 1954.

[39] A. Soylemezoglu, S. Jagannathan, and C. Saygin, “Mahalanobis taguchi system (mts) as a prognostics tool for rolling element bearing failures,” *Journal of Manufacturing Science and Engineering*, vol. 132, no. 5, p. 051014, 2010.

[40] M. Lichman, “UCI machine learning repository,” 2013. [Online]. Available:

[41] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[42] Y. Bulatov, “Notmnist dataset,” *Google (Books/OCR), Tech. Rep.[Online]*, 2011.

[43] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[44] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror, "Result analysis of the nips 2003 feature selection challenge," in *Advances in neural information processing systems*, 2005, pp. 545–552.

[45] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[46] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[47] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of machine learning research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[48] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *Advances in neural information processing systems*, 2012, pp. 341–349.

[49] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[50] A. Nøkland, "Direct feedback alignment provides learning in deep neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 1037–1045.

[51] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman, "Random synaptic feedback weights support error backpropagation for deep learning," *Nature communications*, vol. 7, p. 13276, 2016.

[52] P. Hall, *The bootstrap and Edgeworth expansion*. Springer Science & Business Media, 2013.

[53] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM review*, vol. 38, no. 1, pp. 49–95, 1996.

[54] S. Wright and J. Nocedal, "Numerical optimization," *Springer Science*, vol. 35, no. 67-68, p. 7, 1999.

[55] M. Sion *et al.*, "On general minimax theorems." *Pacific Journal of mathematics*, vol. 8, no. 1, pp. 171–176, 1958.

# V. DISTRIBUTED MINIMAX LEARNING WITH DEEP SPARSE NEURAL NETWORK WITH APPLICATIONS TO HIGH DIMENSIONAL CLASSIFICATION

R. Krishnan, S. Jagannathan

Department of Electrical & Computer Engineering

Missouri University of Science and Technology

Rolla, Missouri 65409–0050

Email: krm9c, sarangap@mst.edu

V.A. Samaranayake

Department of Mathematics & Statistics

Missouri University of Science and Technology

Rolla, Missouri 65409–0050

Email: vsam@mst.edu

## ABSTRACT

In this paper, a novel distributed learning methodology is introduced for deep neural networks (DNN) while considering classification problems with high dimensional data as an application. To address the challenges posed by high dimensional data, a $L_1$ regularized DNN optimization problem is formulated where optimal sparsity is estimated by designing a two player zero-sum game between the penalty coefficients/sparsity parameters and the DNN weights. To solve this game, a novel learning scheme is introduced where layer wise cost functions are derived at each DNN layer by introducing additional variables. As a result, the output at a given DNN layer depends on the additional variables instead of the previous layer. These layer-wise cost functions are optimized to learn the DNN parameters as well as the sparsity coefficients through the minimization approach. The obtained Nash solution of the game provides optimal sparsity and its compensation through the classifier. The proposed approach is finally implemented in a parallel and distributed environment through a novel computational algorithm and demonstrated both theoretically and using nine data-sets.

## 1. INTRODUCTION

Deep neural networks (DNN) are very efficient in environments where big data-sets are available provided efficient learning schemes for training DNN weights can be designed. Ever since its introduction in the 90s, back-propagation [1] has been the most popular approach for training DNN weights. In back-propagation, a cost function is first defined based on the task at hand and then the weights are trained to minimize this cost function through its gradient. The effectiveness of gradient-based techniques has been well established in the literature [2]. However, in spite of their effectiveness, it has also been demonstrated [2, 3] that such gradient-based approaches can suffer from critical drawbacks such as: (1.) vanishing gradients, where learning signals diminish with the increase in the number of layers in the DNN, (2.) poor solutions and local minima, due to improper initial conditions [2], (3.) lack of a proper mechanism for parallelization across layers and (4.) heuristics associated with $L_1$ regularized DNN-learning procedures.

In this paper, a novel methodology is introduced to address the challenges mentioned above. A distributed learning procedure for learning DNN to address the issue of non-convexity and vanishing gradients. Next, a two player game is introduced to minimize heuristics in hyper-parameter selection and then an efficient algorithm is provided for implementing the proposed learning regime in parallel environments. The details of the proposed methodology is provided below.

The use of alternate activation functions such as relu, leaky relu, drelu, etc[4, 5]; and learning rate schedules [5] are common ways of addressing this issue. However, it has been shown that these approaches [5, 6] do not mitigate the vanishing gradient problem completely and only provide a workaround for satisfactory performance [4]. In addition, non-convexity of the DNN-optimization problem renders the DNN-learning very susceptible to initial conditions [7] thus leading to local minima and poor solutions. A solution to this problem is to introduce sustainable strategies for proper initial conditions, that is weight initialization [8]. However, selecting proper initial conditions is incredibly heuristic and requires a tedious trial and error procedure. Alternatively, several methods [9, 10] have been proposed that convert a non-convex problem into a convex one by either using a smoothing parameter [9] or a predetermined transformation [10]. The drawbacks of these approaches [9, 10] include the need to know the exact transformation or the smoothing parameter.

These drawbacks have motivated further research in alternate non-gradient-based learning frameworks [11, 12, 13, 14]. Target propagation was introduced in [11] where difficult to define targets are introduced at each layer with great computational cost to break the learning process across layers. In addition, error-driven learning approaches were proposed in [12, 15] which suffer from inefficiency while optimizing convolutional neural networks (CNN) as is shown in [12]. In the literature, coordinate descent [13] or alternating minimization approaches [14] are also available that can alleviate the issue of non-convexity. These approaches [13, 14] use different mathematical constructs to split the cost function across layers in DNN. However, these approaches [13, 14] need to maintain auxiliary variables for whole data-set, thus impractical in big data scenarios

The presence of high dimensional data-sets, characterized by (large $n$, large $p$), where $n$ denotes the number of data-points and $p$ the dimension, introduces additional challenges into the learning procedure [16]. As the number of parameters in the DNN scales quickly with dimensionality of data estimating parameters of DNN mapping is intractable because enough data-points are not available. Furthermore, challenges such as noisy dimensions, spurious correlation and incidental endogeneity also introduces estimation errors into the learning problem. Because these errors are due to unwanted dimensions and/or weights [16], removing unnecessary weights in DNN is a necessary and an important issue.

A principled way of removing unnecessary weights is to pose the learning using $L_1$ regularized optimization [17] where the $L_1$ norm of the mapping parameters are minimized along with the cost function. The process is known as introducing sparsity into the model and larger sparsity ensures fewer parameters in the DNN model. The associated drawback is the introduction of estimation errors as large sparsity also reduces the expressive power of DNN [18]. This trade-off is typically controlled by carefully choosing the coefficients for the $L_1$ norm penalty, referred as sparsity parameters. Since, it is impossible to manually choose correct values for these penalties in a DNN architecture thus most practitioners use constant values for the penalties. Furthermore, popular methods including grid search[19] and cross-validation approaches [19] are also used towards this end but the presence of heuristics makes these approaches very tedious.

In addition, several attempts have been made to parallelize [20, 21, 22] the learning of DNN across data-points. However, for large models with large number of parameters, a centralized learning problem is still difficult due to the non-convexity of cost function. Furthermore, approaches such as elastic averaging stochastic gradient descent [23] fail when the number of batches in the data becomes large because the parameters have a tendency to converge to zero.

In light of these issues, a principled methodology is introduced in this paper to address challenges in terms of : (1) the non-convexity of cost function; (2) heuristics in the selection of sparsity parameters for $L_1$ regularized optimization and (3) lack of any structured approaches that parallelize over layers in deep NN.

To minimize heuristics, optimal sparsity is estimated by designing a two player zero-sum game between the penalty coefficients/sparsity parameters and the DNN weights. In this game, the first player, the sparsity parameters aim to increase sparsity in the DNN while the second player aims to improve the performance of the DNN classifier. In contrast with cross-validation and grid-search based approaches [19], the proposed approach does not require an intensive search process to select the sparsity parameters.

To address the non-convexity problem, layer wise cost functions are derived at each DNN layer by introducing additional variables such that each of these layer-wise cost function is convex and can be easily optimized. The additional variables are introduced in such a way that the output at a given DNN layer depends on the additional variables instead of the previous layer. Subsequently, these layer-wise cost functions are independently optimized to learn the additional variables, the sparsity parameters and the DNN weights at each layer. The convexity of the optimization problem is also demonstrated theoretically. Finally, the existence of the Nash solution for the game with the additional variables is theoretically demonstrated even when the number of data-batches are large.

As the optimization problem with additional variables is convex, the issues due to non-convexity of cost function are not observed in contrast with [7] and other SGD-based approaches [24]. Though, our approach appears similar to [10] in terms of defining a transformation of the cost function, it is still different due to extra variables instead of pre-determined transformation. Finally, our approach is similar in spirit to [13, 14] with key differences that include (1.) a variable for every data-point in the data-set is not maintained; and (2.) our approach can be generalized to architectures such as CNN in order to work with images efficiently.

Finally, since each of these layer-wise problems are independent, the methodology provides opportunities for high scalability and parallelization. To enable such learning, within the proposed approach a novel implementation algorithm is introduced that enables parallelization across both the data-batches and the layers in the DNN. In contrast with [20, 21, 22], the cost of communicating weights between different segments of the DNN optimization problem is substantially less compared to [20, 21, 22]. The benefits of this approach are demonstrated in simulation on a total of eight data-set where many of them belong to big data. The contributions of this paper include:

1. Introduction of a novel distributed learning approach for DNN. Here, a highly nonlinear and non-convex optimization problem is transformed into simpler nonlinear least square problems.
2. Development of a two player game for mitigating the impact of high dimensionality of data in the model. Through this two player game, sparsity is brought into the learning phase and the impact of heuristics is minimized in contrast with [17, 18].
3. Derivation of theoretical results for the proposed approach.
4. Demonstration of the proposed approach in simulation involving the use of benchmarking data-sets.
5. Development of a parallelized implementation algorithm of the proposed approach.

In the next section, the preliminaries and subsequent notations are discussed. Next the proposed approach is introduced in section II, with section III discussing the results, followed by conclusions in Section IV.

## 2. BACKGROUND AND PRELIMINARIES

The basic notations for the paper are summarized in Table 1. With these notations, consider any manufacturing process where the data at an instance is denoted by $\boldsymbol{x}$. Next, consider the problem of classification where the objective is to determine, whether $\boldsymbol{x}$ belongs to one of the $\mathcal{F}$ categories or not. To predict these categories, the main idea is to estimate a map $\phi(.)$, referred to as classifier, that transforms the data-point $\boldsymbol{x}$ into a probability space where the magnitude of $\phi(\boldsymbol{x})$ indicates the membership of $\boldsymbol{x}$ to different categories.

Membership values signify the probability of $\boldsymbol{x}$ belonging to any particular category. To learn $\phi(.)$ one usually assumes that a data-set $\boldsymbol{\mathcal{X}}$ with labels $\boldsymbol{\mathcal{Y}}$ is available and it represents all the necessary information. The overall learning then involves approximating $\phi(.)$ through a parametric map such as neural networks, spline functions, etc [19].

Table 1. Notations

| Notation | Meaning |
|---|---|
| $\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}}$ | Data-set, Labels |
| $\mathcal{F}$ | Number of faults/classes in the system. |
| $(n, p)$ | Number of sample points in the training set, the dimensionality of data |
| $\mathbb{R}$ and $\mathbb{N}$ | The set of real and natural numbers. |
| $(.)^{(i)}$ | Number of layers in the DNN |
| $P(\hat{\boldsymbol{W}}^{(i)})$ | The penalty function to introduce sparsity |
| $P_1(\hat{\boldsymbol{\lambda}}^{(i)})$ | Function signifying magnitude of the sparsity parameter |
| $\psi()$ | The penalty function to enforce constraints |

In this paper, DNN is chosen to be the parametric map of choice and each training instant is denoted as $k$ with $\boldsymbol{x}_k$ denoting the data-sample from $\boldsymbol{\mathcal{X}}$ at that instant. Next, we assume that $\phi(.)$ can be approximated using a DNN [25] with an ideal set of weights denoted by $\boldsymbol{\theta} = [\boldsymbol{W}^{(1)} \cdots \boldsymbol{W}^{(d)}]^T$ and one may write the ideal parametric map as

$$\begin{aligned}\phi(\boldsymbol{x}_k) &= \boldsymbol{y}(\boldsymbol{x}_k; \theta) + \varepsilon \\ &= f^{(d)}(\boldsymbol{W}^{(d)} \cdots (f^{(1)}(\boldsymbol{W}^{(1)}(\boldsymbol{x}_k))))) + \varepsilon,\end{aligned}$$

where $\boldsymbol{y} \in \mathbb{R}^{\mathcal{F} \times 1}$ represents the ideal output/labels for $\phi(.)$ and the target weights includes the layer-wise bias. Furthermore, $\varepsilon$ is the approximation error and is dependent on the chosen capacity of the DNN[26]. As a consequence, it is assumed constant and bounded through out the estimation procedure. Let the variable $d$ denotes the total number of layers in the DNN and the estimated map at any training instant $k$ be denoted as $\hat{\boldsymbol{y}}(\boldsymbol{x}_k)$. With estimated weights denoted as $\hat{\boldsymbol{\theta}} = [\hat{\boldsymbol{W}}^{(1)} \cdots \hat{\boldsymbol{W}}^{(d)}]^T$, one may write the estimated map as

$$\hat{\boldsymbol{y}}_k(\boldsymbol{x}_k) = \hat{\boldsymbol{y}}_k(\boldsymbol{x}_k; \hat{\boldsymbol{\theta}}) = f^{(d)}(\hat{\boldsymbol{W}}^{(d)} \cdots (f^{(1)}(\hat{\boldsymbol{W}}^{(1)} \boldsymbol{x}_k)))). \tag{1}$$

It can be observed that the estimated map is dependent on the training instances but the ideal map is independent of $k$. To solve this problem, we would usually minimize the difference between the labels(ideal parametric map) and the DNN output(estimated parametric map) at any training instance $k$ such that the difference goes to zero as $k \rightarrow \infty$.

Formally, we define a loss function $\ell(\boldsymbol{y}, \hat{\boldsymbol{y}}(k))$ such that it represents the difference between the labels and the output of the NN and minimize it in the learning procedure. Since we cannot work with the whole data-set at one, we will work with a sample of data and calculate the expected value of loss at any training instance. Let $E[.]$ denote the expectation operator and write the overall cost $\acute{J}$ as the expected value of the loss to get

$$\acute{J} = \quad E_{\forall \boldsymbol{x}_k \in \boldsymbol{\mathcal{X}}, \boldsymbol{y} \in \boldsymbol{\mathcal{Y}}}[\ell(\boldsymbol{y}, \hat{\boldsymbol{y}}(\boldsymbol{x}_k))]. \tag{2}$$

Note that we replace the ideal map with labels from the data-set while calculating the expected values. For brevity of notation, we will suppress the sub-scripts for the expected values from hereon. Formally, the learning objective is then obtained as the minimization of this cost function $\acute{J}$ to estimate the weights such that

$$\boldsymbol{\theta}^* = \underset{\hat{\theta} \in \Omega}{\arg\min}\, \acute{J}(k), \tag{3}$$

where $\Omega$ represents the parameter space.

**Remark 30** *The term $\ell(\boldsymbol{y}, \hat{\boldsymbol{y}}(k))$ can denote any generic loss function where examples include quadratic ($L_2$ norm-based) loss and cross-entropy loss function[19].*

With this setup, the proposed approach is described next.

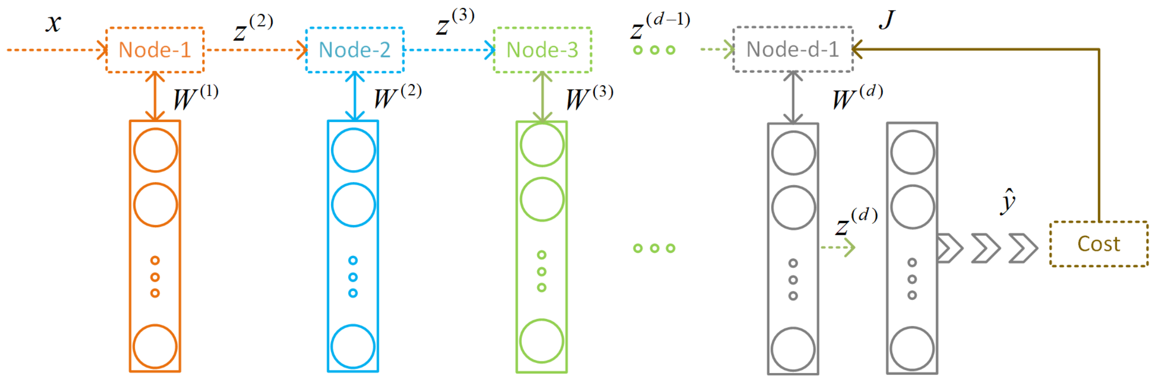


Figure 1. Illustration for the proposed approach.

## 3. OVERALL METHODOLOGY

In this section, we first introduce the $L_1$ regularized optimization framework. Next, a distributed learning approach is demonstrated to solve the optimization problem for which a pictorial representation is given in Figure 1 Then, the details of the two player game are presented where the sparsity parameter is introduced as player in the optimization problem. Finally, the overall approach is theoretically analyzed.

Many challenges are observed while minimizing a cost function in the presence of high dimensional data-sets. Typically, the size of DNN increases with the number of dimensions in the data. If $n$ is not greater than $p$ then the system of equations to solve (to estimate the weights that approximate the map $\phi(.)$) is under-determined. This situation leads to a lot of unwanted parameters in the DNN due to which the DNN would overfit on the data and result in increased estimation errors [19]. In these scenarios, it is preferable to reduces the number of parameters in the DNN during the training phase. Typically, the process involves introducing sparsity into the DNN using a $L_1$ norm constraint on the weights. Let $P(\hat{\mathbf{W}}^{(i)})$ denote the constraint that is applied on the weights and write

$$\acute{J}(\hat{\mathbf{y}};\hat{\boldsymbol{\theta}},\hat{\boldsymbol{\lambda}}) = E[\ell(\mathbf{y},\hat{\mathbf{y}}(k))] + \sum_{i=1}^{d} \hat{\boldsymbol{\lambda}}^{(i)} P(\hat{\mathbf{W}}^{(i)}). \tag{4}$$

If the penalty function is chosen as the $L_1$ norm then we get $P(\hat{\boldsymbol{W}}^{(i)}) = |\hat{\boldsymbol{W}}^{(i)}|$ which denotes our $L_1$ regularized scenario. The vector $\hat{\boldsymbol{\lambda}} = [\hat{\boldsymbol{\lambda}}^{(i)} \cdots \quad \cdots \hat{\boldsymbol{\lambda}}^{(d)}]$ is also referred to as the sparsity parameters and is generally chosen using the experience of the practitioner. In $L_1$ norm based optimization, it is fairly traditional to set the value of sparsity parameters for each layer to be equal to some constant that is $\hat{\boldsymbol{\lambda}}^{(i)} \quad \cdots = \cdots \quad \hat{\boldsymbol{\lambda}}^{(d)} = c$ [17, 27, 28]. The objective then is to optimize Eq. (4) to learn the weights and the overall learning problem is given by Eq. (3). To optimize the proposed problem, a distributed learning approach is developed next.

## 3.1. DISTRIBUTED LEARNING APPROACH

It was mentioned before in the introduction that there are several drawbacks with a traditional gradient based approach when the function to optimize has functional nesting(the form denoted in Eq. (1)). In DNN-terminology, this refers to the presence of multiple layers that leads to the non-convexity which is observed in optimization procedures. To alleviate this issue, the principal idea is to disassociate the learning problem across the layers such that each layer can be optimized independently. We accomplish this by introducing variables, referred as additional variables at each layer in the DNN and optimizing these additional variables along with the weights.

To elaborate, let us start by introducing a variable $\boldsymbol{z}^{(d)}$ to replace the top-most layer in the DNN such that

$$\begin{aligned} \acute{J}(\hat{\boldsymbol{y}}; \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\lambda}}) &= J(\hat{\boldsymbol{y}}(\boldsymbol{z}^{(d)}); \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\lambda}}) \\ &= E[\ell(\boldsymbol{y}, \hat{\boldsymbol{y}}(\boldsymbol{z}^{(d)})] + \sum_{i=1}^{d} \hat{\boldsymbol{\lambda}}_i P(\hat{\boldsymbol{W}}), \end{aligned}$$

where $\hat{\boldsymbol{y}}(\boldsymbol{z}^{(d)}) = f^{(d)}(\boldsymbol{z}^{(d)})$ is the output of the DNN that depends on $\boldsymbol{z}^{(d)}$. It follows that the loss function depends on $\boldsymbol{z}^{(d)}$. The advantage as observed is that the optimization of the cost function does not have any nesting therefore it is equivalent to a one layer NN optimization problem. One layer NN optimization problem is relatively simpler to the original problem and has been well studied in the literature [19]. However, as optimization procedure is now dependent on the additional variables instead of the topmost layer in the DNN, the additional variables must mimic the topmost layer in the DNN. This can be achieved by adding a constraint into the optimization problem such that

$$z^{(d)}(\boldsymbol{x}) = \hat{\boldsymbol{W}}^{(d)} f^{(d-1)}(\boldsymbol{x}).$$

It can be observed that these additional variables are local observers that estimate the information required for learning the weights at layer $d$.

Next, lets introduce another variable $z^{(d-1)}$ that is constrained to mimic the penultimate layer in the DNN such that the topmost layer depends on $z^{(d-1)}$. It follows that the constraint on variable $z^{(d)}$ can be rewritten as

$$z^{(d)} = \hat{\boldsymbol{W}}^{(d)} f^{(d-1)}(z^{(d-1)}).$$

As mentioned earlier, this dependence introduces an additional constraint as

$$z^{(d-1)} = \hat{\boldsymbol{W}}^{(d-1)} f^{(d-2)}(z^{(d-2)}).$$

With this line of thought one may define a variable at each layer of the neural network constrained to mimic all the information prior to that particular layer in the DNN. Due to this, the overall optimization problem can be rewritten as

$$\begin{aligned}
\boldsymbol{\theta}^* = & \operatorname*{arg\,min}_{\hat{\theta}\in\Omega} J(\hat{\boldsymbol{y}}(\boldsymbol{z}^{(d)}); \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\lambda}}) \\
\text{subject to} & \\
\boldsymbol{z}^{(d)} = & \hat{\boldsymbol{W}}^{(d)} f^{(d-1)}(\boldsymbol{z}^{(d-1)}) \\
\boldsymbol{z}^{(d-1)} = & \hat{\boldsymbol{W}}^{(d-1)} f^{(d-2)}(\boldsymbol{z}^{(d-2)}) \\
\vdots & \\
\boldsymbol{z}^{(2)} = & \hat{\boldsymbol{W}}^{(2)} f^{(1)}(\boldsymbol{z}^{(1)}),
\end{aligned}$$

where $\boldsymbol{z}^{(1)}(\boldsymbol{x}) = \hat{\boldsymbol{W}}^{(1)}\boldsymbol{x}$. Observe that each constraint is dependent on the weight, the activation function at a layer and the additional variables corresponding to the layer below. As a result, the optimization problem from Eq. (3) can now be rewritten as shown in Eq. (5). The only difference in between these problems is the presence of additional variables and the constraints.

One might also observe that a straight forward unconstrained DNN optimization problem is being changed to introduce constraints. These constraints also introduce more variables and more parameters that must be optimized. However, the main advantage is the fact that an inherently non-convex optimization problem is converted into a series of sub-problems that can be solved independently. This introduces opportunities for parallelization and efficient scalability.

The constraints introduced into the new problem are already present in a standard DNN implicitly. However, in a traditional case, these constraints generally take the form of "input to each hidden layer in the DNN must be equal to the output of the previous layer" and we do not think of them as constraints. By introducing these additional variables, these constraints become explicit. The advantage as observed is that a optimization problem that depended on nested functional form of DNN does not depend on nested functions anymore. However, this approach requires that the constraints are satisfied.

To enforce these constraints, we introduce a generic penalty function $\psi(.)$ [29] such that

$$\begin{aligned} J(\hat{\boldsymbol{y}};\hat{\boldsymbol{\theta}},\boldsymbol{Z},\hat{\boldsymbol{\lambda}}) &= E[\ell(\boldsymbol{y},\hat{\boldsymbol{y}}(\boldsymbol{z}^{(d)})] + \sum_{i=1}^{d} \lambda_i P(\hat{\boldsymbol{W}}^{(i)}) \\ &+ \sum_{i=2}^{d} \gamma_i \psi(\boldsymbol{z}^{(i)}, \hat{\boldsymbol{W}}^{(d)} f^{(i-1)}(\boldsymbol{z}^{(i-1)})), \end{aligned}$$

where $\boldsymbol{Z} = [\boldsymbol{z}^{(1)} \quad \cdots \quad \boldsymbol{z}^{(d)}]$. Observe that $\gamma_i$ is the penalty coefficient for weight at the layer $i$ and $\psi(.)$ denotes a penalty function of choice. The choice of these penalty coefficients decide the exactness with which these constraints are enforced. In a traditional DNN, these constraints are enforced exactly, which is not the case in this formulation. It follows that at the cost of some imprecision in the constraints at each layer of the DNN, we achieve an optimization problem that is convex and is equivalent to the original optimization problem. The convexity is demonstrated in Lemma 1. Next, we will introduce the zero-sum game.

## 3.2. ZERO-SUM GAME

A main drawback of the learning approach described above is that the extent of sparsity that can be introduced is unknown. Therefore, most practitioners will choose the sparsity parameters through a trial and error process. The process is necessary because large sparsity (more magnitude of the sparsity parameters) reduces the expressive power (capacity) of the model thus increasing the learning cost. With lower capacity, the DNN may fail at estimating the map $\phi(.)$ effectively leading to a violation of the assumption that the approximation error is constant and bounded.

It is therefore preferable to let the learning problem establish a trade-off between the expressiveness (learning cost) and sparsity of the DNN(magnitude of the sparsity parameters). This trade-off is closely tied to the magnitude of the sparsity parameters. Specifically, the larger the magnitude of the sparsity parameter, the greater the sparsity in the weight and lower the expressive power of the DNN. Thus, to manipulate sparsity in the DNN, we manipulate the magnitude of the sparsity parameter. Lets denote this magnitude as $P_2(\hat{\boldsymbol{\lambda}})$. Next a two player game is designed where the objective is to increase $P_2(\hat{\boldsymbol{\lambda}})$ as much as possible.

Formally. define a ratio between the current cost and the norm of the sparsity parameter is defined as $\frac{J(\hat{\boldsymbol{y}};\hat{\boldsymbol{\theta}},\boldsymbol{Z})}{P_2(\hat{\lambda})}$. The ratio formalizes the trade-off between the cost function and sparsity parameters and the objective is to keep this ratio within a threshold $\Gamma$, it follows that $\frac{J(\hat{\boldsymbol{y}};\hat{\boldsymbol{\theta}},\boldsymbol{Z})}{P_2(\hat{\lambda})} \leq \Gamma^2$, where $P_2(\hat{\boldsymbol{\lambda}})$ is indicative of the magnitude. With this being the objective, we can rewrite the cost function by including this constraint to get

$$\begin{aligned} H(\hat{\boldsymbol{y}};\hat{\boldsymbol{\theta}},\boldsymbol{Z},\boldsymbol{\lambda}) = E[\ell(\boldsymbol{y},\hat{\boldsymbol{y}}(\boldsymbol{z}^{(d)})]+ &\sum_{i=1}^{d} \lambda_i P(\hat{\boldsymbol{W}}^{(i)}) \\ +\sum_{i=2}^{d} \gamma_i \psi(\boldsymbol{z}^{(i)},\hat{\boldsymbol{W}}^{(d)} f^{(i-1)}(\boldsymbol{z}^{(i-1)}))- &\sum_{i=1}^{d} \Gamma^2 P_2(\hat{\boldsymbol{\lambda}}). \end{aligned}$$

In the cost function described in Eq. (5), sparsity is introduced into the learning problem through the second term. On the other hand, the fourth term is representative of the magnitude of $\hat{\boldsymbol{\lambda}}$. As a consequence, we observe that the objective in Eq. (5) can be restated in game theoretic parlance as the objective of improving the performance of the classifier by reducing the overall error while increasing the amount of sparsity in the model. Note that learning cost and the sparsity of the DNN are tied through the second term. It follows that with an increase in the magnitude of the sparsity parameters the cost increases. Similarly, with an reduction in the cost value, the sparsity parameters are effected.

**Remark 31** *The two player game introduced here may not be restricted to the choice of penalty parameters for the $L_1$. The same setup can be used to solve for other user-defined parameters in any optimization. For example, the parameters of $\gamma_i$ in the proposed learning regime can also be introduced through the setup described above.*

In contrast with most traditional approaches where the sparsity parameters must be chosen using the practitioners experience, grid-search or cross validation, the parameters are obtained through the learning process. Finally, optimization problem in Eq. (3) can be restated as follows

$$(\hat{\boldsymbol{\theta}}^*, \hat{\boldsymbol{Z}}, \hat{\boldsymbol{\lambda}}^*) = \underset{\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{Z}} \in \Omega, \Omega_Z}{\arg\min} \; \underset{\hat{\boldsymbol{\lambda}} \in \Omega_\lambda}{\arg\max} \, H(\hat{\boldsymbol{y}}; \hat{\boldsymbol{\theta}}, \boldsymbol{Z}, \boldsymbol{\lambda}). \tag{5}$$

where $\Omega, \Omega_Z, \Omega_\lambda$ are the parameter space for the different variables and $\hat{\boldsymbol{Z}} = [z^2 \cdots z^d]$. The stationary conditions for the optimization can be written as

$$\begin{aligned} \boldsymbol{Z}^* &= \arg\min H \\ \hat{\boldsymbol{\theta}}^* &= \arg\min H, \\ \boldsymbol{\lambda}^* &= \arg\max H, \end{aligned}$$

which results in the parameter updates at each iteration $k$ using gradient descent as

$$\begin{aligned} \boldsymbol{Z}(k+1) &= \boldsymbol{Z}(k) - \eta_2 \Delta_{\boldsymbol{Z}} H \\ \hat{\boldsymbol{\theta}}(k+1) &= \hat{\boldsymbol{\theta}}(k) - \eta_1 \Delta_{\hat{\boldsymbol{\theta}}} H, \\ \hat{\boldsymbol{\lambda}}(k+1) &= \hat{\boldsymbol{\lambda}}(k) + \eta_3 \Delta_{\hat{\boldsymbol{\lambda}}} H, \end{aligned}$$

where $\eta_1, \eta_2, \eta_3$ are the specified learning rate.

Once the optimization problem has been solved and the optimum DNN has been obtained, one generally, applies the DNN model on new data. Note that, at the end of the optimization process, the additional variables mimic the output at the hidden layers in the DNN such that all the constraints are satisfied.

Since, this case is equivalent to the case when the additional variables are not present, one can replace all the variables with a traditional DNN, where the weights are frozen at the optimized values. This replacement would just imply that all the constraints are satisfied. A pictorial representation of the proposed approach is shown in Figure 1.

In the next section, some key points about the methodology are analyzed.

## 3.3. ANALYSIS

One may observe that the proposed formulation in Eq. (5) represents a mixed strategy two player game. Therefore, a saddle point solution is not guaranteed and it must be demonstrated [31]. To this end, we would first show that $H$ satisfies the conditions under which a saddle point solution is possible. We demonstrate that the conditions are true even when the number of layers in the neural network as well as the number of neurons at every layer tend to infinity. This is shown in Lemma 7. Next, under the strategies chosen by the two players, it will be shown in Theorem 4 that proposed formulation would achieve Nash equilibrium. With a little abuse of notation, we will redefine $\hat{\boldsymbol{\theta}}$ to include all the $\hat{\boldsymbol{Z}}$. Therefore, $(\hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{Z}}) = (\hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\theta}})$ and whenever we state parameters of the neural networks, we mean the both the weights and the additional variables.

Before stating the results, we will first choose cross entropy as the loss function and choose log barrier for penalty. Thus, we will rewrite the complete cost function as

$$H = H(\hat{\boldsymbol{y}}; \hat{\boldsymbol{\theta}}, \boldsymbol{Z}, \boldsymbol{\lambda}) = E[\boldsymbol{y}^T log(\hat{\boldsymbol{y}}(\boldsymbol{z}^{(d)}))^{-1}] + \sum_{i=1}^{d} \lambda_i |\hat{\boldsymbol{W}}^{(i)}|$$
$$+ \sum_{i=2}^{d} \gamma_i log(1 - \frac{(\boldsymbol{z}^{(i)} - \hat{\boldsymbol{W}}^{(i)} f^{(i-1)}(\boldsymbol{z}^{(i-1)}))}{\gamma_i})^{-1} - \sum_{i=1}^{d} \Gamma^2 \lambda_i^2.$$

With this setup, we introduce our first lemma as follows.

**Lemma 7** *Let H be defined as shown in Eq.* (5) *and let the solution pair be given as* $(\boldsymbol{\lambda}^*, \boldsymbol{\theta}^*)$ *and let the parameter set be given as* $(\Omega_{\lambda^*}, \Omega)$. *Consider the case when the number of layers in the DNN tend to infinity and let* $\sum_{i=2}^{d} \gamma_i = L$ *and* $\sum_{i=1}^{d} \lambda_i = L$ *where* $L \in \mathbb{R}$. *It follows that H is convex with respect to the parameters of the neural network. Similarly, H is concave with respect to the sparsity parameters. Proof : See Appendix*

With the above result being true, it can be shown the proposed formulation can achieve a saddle point.

**Theorem 4** *Let Lemma 7 be valid and assume that the parameters sets are compact, it is concluded that the two player zero-sum game represented by H has a saddle point denoted by* $(\boldsymbol{\lambda}^*, \boldsymbol{\theta}^*)$. *Proof: See Appendix.*

The following observations can now be made as a consequence of the above theorem.

**Remark 32**

1. *The conditions that the number of layers in the DNN and the number of neurons in each layer tends to infinity reflects the scenario when the number of dimensions are so large that one needs an exceptionally large DNN to analyze it.*
2. *Even though, the result above is demonstrated for cross entropy cost function, it is true for any function that is convex which includes the least square cost function.*
3. *The proof also depends on the convexity of the norm in large dimensions. It has been shown in the literature [30] that convex functions hold convexity in high dimensions. See [30] and the references therein.*
4. *The compactness of the parameter set is important for the validity of Theorem 4 and Lemma 7. In the context of a game it implies that there are finite number of strategies for the two players to take.*

By introducing additional variables, the proposed approach lends itself to parallelization because all the layers in the DNN are not connected as is traditional with deep neural networks. Furthermore because the additional variables $Z$ replicate the data at the hidden layers in the DNN. The convergence of the Zs before the weight updates is important and the learning rate for updating $\boldsymbol{Z}$ is chosen to be larger than the learning rule for the weights.

**Remark 33** 1. *Note that the variable $z^{(d)}$ acts as a proxy for the top-most layer. Due to the presence of this additional variables. All the information required to update weights at any layer may be estimated locally. Therefore, the learning of weights at any layer does not depend on anything other than the current layer.*

2. *Furthermore, the number of updates for the additional variables can also be chosen. A thumb rule is to update the $\mathbf{Z}$ at least 10 times prior to the update of the weights. This is to ensure that the additional variables converge faster than the weights.*

**Algorithm 7** Algorithm For the Proposed Approach

```
1: for each epoch $\boldsymbol{k}$ do
2:     Input: $\boldsymbol{X}$
3:     for each batch in $\boldsymbol{X}$ do
4:         Compute Cost $\boldsymbol{H}$
5:         for $j = 1 \rightarrow T$ do
6:             Expectation Step:
                   $\hat{\boldsymbol{Z}}(k+1, j+1) = \hat{\boldsymbol{Z}}(k+1, j) - \eta_1 \Delta_{\hat{\boldsymbol{Z}}(k+1,j)} H$
7:         end for
8:         Learning Step:
                   $\hat{\boldsymbol{\theta}}(k+1) = \hat{\boldsymbol{\theta}}(k) - \eta_2 \Delta_{\hat{\boldsymbol{\theta}}(k)} H$
                   $\hat{\boldsymbol{\lambda}}(k+1) = \hat{\boldsymbol{\lambda}}(k) + \eta_3 \Delta_{\hat{\boldsymbol{\lambda}}(k)} H$
9:     end for
10: end for
```

These benefits demonstrate the possibility of a parallel implementation, one such algorithm is presented in the next subsection.

## 3.4. PARALLEL IMPLEMENTATION

Traditionally, weight updates at any particular layer requires the use of all the layers above it. However, such dependency does not exist with the proposed approach. One can observe this by decomposing the total cost into one at each layer. To achieve this layer wise cost, one may rewrite Eq. (6) as

$$
\begin{aligned}
H = E[\boldsymbol{y}^T log(\hat{\boldsymbol{y}}(\boldsymbol{z}^{(d)}))^{-1}] \\
+\Bigg[\gamma_d log(1 - \frac{(\boldsymbol{z}^{(d)} - \hat{\boldsymbol{W}}^{(d)} f^{(d)}(\boldsymbol{z}^{(d-1)}))}{\gamma_d})^{-1} \\
+\lambda_d|\hat{\boldsymbol{W}}^{(d)}| - \Gamma^2\lambda_d^2\Bigg] \\
+\Bigg[\gamma_{d-1} log(1 - \frac{(\boldsymbol{z}^{(d-1)} - \hat{\boldsymbol{W}}^{(d-1)} f^{(d-2)}(\boldsymbol{z}^{(d-2)}))}{\gamma_{d-1}})^{-1} \\
+\lambda_{d-1}|\hat{\boldsymbol{W}}^{(d-1)}| - \Gamma^2\lambda_{d-1}^2\Bigg] \\
+\cdots \\
+\Bigg[\gamma_2 log(1 - \frac{(\boldsymbol{z}^{(2)} - \hat{\boldsymbol{W}}^{(2)} f^{(1)}(\boldsymbol{z}^{(1)}))}{\gamma_2})^{-1} \\
+\lambda_2|\hat{\boldsymbol{W}}^{(2)}| - \Gamma^2\lambda_2^2\lambda_1 \\
+|\hat{\boldsymbol{W}}^{(1)}| - \Gamma^2\lambda_1^2\Bigg].
\end{aligned}
$$

Note that the first three terms in Eq. (6) involves terms with $\boldsymbol{z}^{(d)}$ and $\hat{\boldsymbol{W}}^{(d)}$ and forms the cost function for the top-most layer that is $H^{(d)}$. Similarly, the last three terms are involved with variables $\boldsymbol{z}^{(1)}$, $\hat{\boldsymbol{W}}^{(2)}$ and $\hat{\boldsymbol{W}}^{(1)}$ that are combined to get the cost $H^{(1)}$. With these observation, the series of layer-wise costs can be obtained from Eq. (6) as

$$H^{(i)} = \begin{cases} \Big[ E[\boldsymbol{y}^T log(\hat{\boldsymbol{y}}(\boldsymbol{z}^{(d)}))^{-1}] \\ +\gamma_d log(1 - \frac{(\boldsymbol{z}^{(d)} - \hat{\boldsymbol{W}}^{(d)} f^{(d)}(\boldsymbol{z}^{(d-1)}))}{\gamma_d})^{-1} \\ +\lambda_d |\hat{\boldsymbol{W}}^{(d)}| - \Gamma^2 \lambda_d^2 \Big] & i = d \\ \\ \Big[ + \gamma_{i+1} log(1 - \frac{(\boldsymbol{z}^{(i+1)} - \hat{\boldsymbol{W}}^{(i+1)} f^{(i)}(\boldsymbol{z}^{(i)}))}{\gamma_{i+1}})^{-1} \\ +\gamma_i log(1 - \frac{(\boldsymbol{z}^{(i)} - \hat{\boldsymbol{W}}^{(i)} f^{(i-1)}(\boldsymbol{z}^{(i-1)}))}{\gamma_i})^{-1} \\ \lambda_i |\hat{\boldsymbol{W}}^{(i)}| - \Gamma^2 \lambda_2^i \Big] & i < d \\ \\ \Big[ \gamma_2 log(1 - \frac{(\boldsymbol{z}^{(2)} - \hat{\boldsymbol{W}}^{(2)} f^{(1)}(\boldsymbol{z}^{(1)}))}{\gamma_2})^{-1} \\ +\lambda_2 |\hat{\boldsymbol{W}}^{(2)}| - \Gamma^2 \lambda_2^2 \\ +\lambda_1 |\hat{\boldsymbol{W}}^{(1)}| - \Gamma^2 \lambda_1^2 \Big] & i = 1. \end{cases}$$

It can be observed that the derivatives for the variables $\boldsymbol{z}^{(i)}$ and $\hat{\boldsymbol{W}}^{(i)}$ depend on the cost $H^{(i)}$ and thus the updates for these variables can be achieved from these individual cost. With these features, we can design an algorithm for parallel implementation where the parallelization is across the number of layers in the DNN as well as the data-set. This algorithm is described as follows.

Using the concept of multi-processing, the proposed learning regime can be implemented in a parallel fashion. Consider therefore that every hidden layer in the DNN or a group of such hidden layers in the DNN are combined to form a process and each process shall maintain its own set of variables.

Let $\boldsymbol{j}$ denotes a batch of data in the data-set $\boldsymbol{\mathcal{X}}$ and let $\boldsymbol{i}$ denote a process and has variables $\mathcal{Z}^{(i)} = [z^{(i,l)}(j), l = 1, \cdots, L]$, $\hat{\boldsymbol{\mathcal{W}}}^{(i)} = [\hat{\boldsymbol{W}}^{(i,l)}(j), l = 1, \cdots, L]$ and $\hat{\boldsymbol{\Lambda}}^{(i)} = [\hat{\lambda}^{(i,l)}(j), l = 1, \cdots, L]$. Here, $L$ refers to the number of layers in a group. Learning is performed by computing gradients $\Delta_{\hat{\mathcal{Z}}^{(i)}} H^{(i)}, \Delta_{\hat{\boldsymbol{\mathcal{W}}}^{(i)}} H^{(i)}$ respectively on the local cost denoted as $H^{(i)}$.

The algorithm is given in Algorithm 8. Observe that for each process $\boldsymbol{i}$, the variables $\hat{\mathcal{Z}}^{(i-1)}(j)$, $\hat{\boldsymbol{\mathcal{W}}}^{(i-1)}(j)$ and $\hat{\boldsymbol{\Lambda}}^{(i-1)}(j)$ are inputs from the process $i - 1$. Next, the variables $\hat{\mathcal{Z}}^{(i+1)}(j - 1)$, $\hat{\boldsymbol{\mathcal{W}}}^{(i+1)}(j - 1)$ and $\hat{\boldsymbol{\Lambda}}^{(i+1)}(j - 1)$, are obtained from the process $i + 1$. The resultant variables are obtained as $\hat{\mathcal{Z}}^{(i-1)}(j)$, $\hat{\boldsymbol{\mathcal{W}}}^{(i-1)}(j)$ and $\hat{\boldsymbol{\Lambda}}^{(i-1)}(j)$ and these variables represent the updated values from the current batch of data. As a consequence, every process results in the variables $\hat{\mathcal{Z}}^{(i)}(j)$, $\hat{\boldsymbol{\mathcal{W}}}^{(i)}(j)$ and $\hat{\boldsymbol{\Lambda}}^{(i)}(j)$ as outputs.

Note from the algorithm that when the process $(i)$ is being optimized, the processes $(i - 1)$ can update itself on the next batch of data. This presents a unique algorithm to implement this methodology on streaming data with efficient parallelization over data-batches. Since a process is defined for each layer/ a group of layers in the DNN, we can achieve parallelization across different layers in the DNN.

Finally, the advantages and disadvantages of the proposed work are summarized

**Remark 34** *The following advantages are achieved due to the proposed methodology*

1. *In contrast with [24], the issue of vanishing gradients is not present because a cost function is obtained for each layer.*

2. *In contrast with other distributed learning mechanisms [22], the cost of communication between different nodes is reduced significantly less compared to [22]. Furthermore, in contrast with [21, 20], the proposed approach is designed to parallelize across different layers in the DNN as well as different data-samples.*

3. *As demonstrated before in Lemma 1, the redefined cost function is convex and therefore enjoys the theoretical support of convex optimization theory. As a result, a highly non-convex nested problem has been converted into a convex optimization problem that can be solved with theoretical guarantees.*

4. *The major disadvantage of the proposed approach is that these use of these additional variables introduce a computational load. However, with an appropriate parallelized algorithm, the impact of such a load can be minimized. One such algorithm was introduced in this paper.*

5. *One big disadvantage is that due to use of these additional variables, we have introduced some extra tuning parameters that must be chosen for optimality. However, certain thumb rules for the choice of these parameters can be summarized and are shown in Remark 33.*

Finally, a set of algorithms for implementation purposed are given in Algorithm 7 and 8. The results are provided next.

**Algorithm 8** Parallel implementation for DNN

```
1:  for each process $\boldsymbol{i}$ do
2:      for each epoch $\boldsymbol{k}$ do
3:          Input: $\boldsymbol{\mathcal{X}}$
4:          for each batch $\boldsymbol{j}$ in $\boldsymbol{\mathcal{X}}$ do
5:              Input: $\hat{\mathcal{Z}}^{(i-1)}(j), \hat{\boldsymbol{\mathcal{W}}}^{(i-1)}(j)$
6:              Input: $\hat{\mathcal{Z}}^{(i+1)}(j-1), \hat{\boldsymbol{\mathcal{W}}}^{(i+1)}(j-1)$
7:              Compute Local Cost $\boldsymbol{H}^{(i)}$
8:              Expectation Step:
9:              for $m = 1 \to K$ do
                    $$\hat{\mathcal{Z}}^{(i)}(j) = \hat{\mathcal{Z}}^{(i)}(j-1) - \eta_2 \Delta_{\hat{\mathcal{Z}}^{(i)}} \boldsymbol{H}^{(i)}$$
10:             end for
11:             Learning Step:
                    $$\hat{\boldsymbol{\mathcal{W}}}^{(i)}(j) = \hat{\boldsymbol{\mathcal{W}}}^{(i)}(j-1) - \eta_1 \Delta_{\hat{\boldsymbol{\mathcal{W}}}^{(i)}} \boldsymbol{H}^{(i)}$$
                    $$\hat{\boldsymbol{\Lambda}}_i(j) = \hat{\boldsymbol{\Lambda}}_i(j-1) - \eta_3 \Delta_{\hat{\boldsymbol{\Lambda}}_i} \boldsymbol{H}^{(i)}$$
12:         end for
13:     end for
14: end for
```

## 4. RESULT AND DISCUSSION

The details of all the data-sets are illustrated below:

1. **Arcene** data-set, is a small data-set of only 200 data points with twenty thousand dimensions that is used for a two-class classification of cancer vs normal from mass-spectrometric data. The data-set has been divided into 50% split for training and testing, i.e. 100 points each.

2. **CIFAR-10** contains a total of 60,000 colored, 32X32 dimensional images for 10 classes, with 6,000 images for each class. The data-set has been split into 50,000 images for training and 10,000 images for testing.

3. **Gas Sensor Array** This archive contains 239000 measurements from 16 chemical sensors utilized in simulations for drift compensation in a discrimination task of 8 gases at various levels of concentrations. A total of 72 features are extracted from this data-set and presents and 8 class discrimination problem.

4. **Gisette** data-set is a 7,000 data-points data-set, used for a two-class hand-written digits classification, to differentiate the digits 4 and 9. The training set contains 6000 data-points with five thousand dimensions, while testing set contains 1000 data-points.

5. **Madelon** is a data-set of 2600 data-points with 500 dimensions introduced in the NIPS 2003 feature selection challenge. It is used for two-class classification and has an approximate split of 80% for training and rest for testing.

6. **MNIST** is a data-set composed of 28X28 gray-scale images of handwritten digits. The data-set is composed of 55,000 images for training and 10,000 images for testing, i.e about 82% of data-set is used for training and 18% for testing.

7. **NotMNIST** data-set is similar to the MNIST data-set, but instead of the digits, the 10 classification is for the English letters A-J. The data-set contains 22,000 data-points and a split of 20,000 points in the training and rest in testing.

8. **Rolling** Two roller bearings were installed on a CNC machine that was operated at 1600 rpm. Vibration and temperature data from both bearings were sampled and recorded continuously at a rate of 10 kHz and eleven features were extracted. This data-set presents a four class fault diagnostics problem.

9. **Sensorless** is a large data-set whose features have been extracted from electric current drive signals, for a 11 class classification of the different conditions experienced in it. The training-testing split is 55-45%.

First, the effect of the game on deciding sparsity is discussed.

## 4.1. PERFORMANCE ANALYSIS - SPARSITY CONTROL

All results in this section are summarized using mean accuracy and standard deviation over hundred iterations. For the purpose of analysis, all nonlinear transformations are chosen to be tanh. The update steps for $Z$ is chosen as 150. The learning rates are chosen as 0.001 and the results are tabulated in Table 2. First, accuracies for the scenario with sparsity regularization is compared with the scenario when no sparsity regularization. With

Table 2. Accuracies for the different data-sets with the proposed method (with sparsity control) and method without sparsity control. The last two column represent the results with and without the Z's

| **Data-set** | **With Regularization** | **Without Regularization** | **Without Z** | **With Z** |
|---|---|---|---|---|
| Arcene | 0.86 | 0.73 | 84 | **86** |
| Dexter | 0.99 | 0.69 | 98 | **99** |
| Gisette | 0.99 | 0.71 | 97 | **99** |
| Madelon | 0.85 | 0.79 | 85 | **85** |
| MNIST | 0.94 | 0.89 | 83 | **94** |

an increase in the number of dimensions in the data, the difference in accuracy increases. with sparsity regularization performing better. The proposed methodology is optimal when the total number of data-points is less than the total dimensions in the data as observed from Arcene, Gisette and Dexter data-set. Next, the sparsity parameters are fixed and the impact of Zs on the overall approach is tested.

## 4.2. RESULTS AND DISCUSSION - PARALLEL IMPLEMENTATION

### 4.2.1. Implementation.

We implement the proposed algorithm using the Python library *multiprocessing*. In this procedure, two processes are constructed and the total number of layers in the DNN are divided into these two process such that each process contains an equal number of layers. The communication between the two processes is performed through file read and write. This two-process parallel implementation is used for all the experiments that are discussed in this section.

Table 3. Accuracy's for different parameter changes

(a)

| $f^{(i)}$ | **Accuracy** |
|---|---|
| tanh | 94.19 |
| relu | 87.17 |
| sigmoid | 74.27 |

(b)

| $\lambda$ | **Accuracy** |
|---|---|
| 1 | 94.19 |
| 5 | 93.83 |
| 10 | 94.19 |
| 15 | 94.33 |
| 50 | 94.33 |

(c)

| $\eta_1$ | **Accuracy** |
|---|---|
| 0.1 | 22.13 |
| 0.01 | 27.12 |
| 0.001 | 12.47 |
| 0.0001 | 94.19 |

(d)

| $\eta_2$ | **Accuracy** |
|---|---|
| 0.01 | 12.61 |
| 0.1 | 94.19 |
| 0.5 | 93.65 |
| 0.8 | 94.17 |
| 1 | 93.74 |

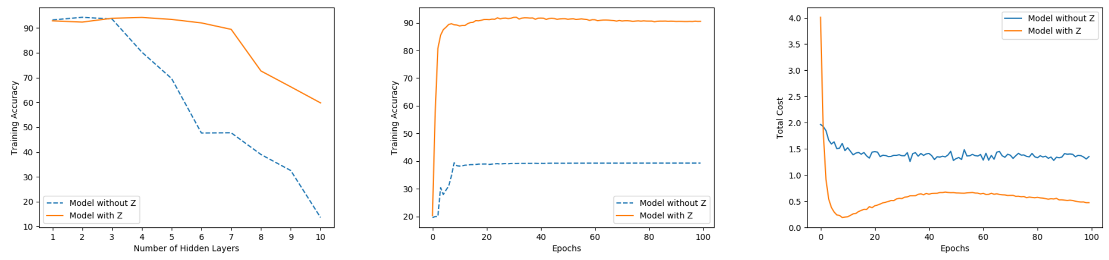
The performance is first demonstrated on the Mnist data-set.

Table 4. % accuracies for the various data-sets with the proposed framework and four different learning algorithms with generalization error in the parenthesis.

| Datasets | $p$ | $n$ | $\mathcal{F}$ | Model Parameters | **DFA** | **FA** | **SGD** | **EDL** |
|---|---|---|---|---|---|---|---|---|
| **Rolling** [32] | 11 | 35000 | 4 | $\alpha = 0.01, b = 100, d = 7, hln = 100,$ $\lambda = 0.001, n_{neigh} = 2, \rho = 1$ | 99(0.0001) | 99(0.0001) | 99(0.6) | 99(0.0) |
| **Sensorless** [33] | 48 | 78000 | 11 | $\alpha = 0.01, b = 100, d = 8, hln = 100,$ $\lambda = 0.001, n_{neigh} = 2, \rho = 1$ | 94(0.008) | 94(0.008) | 95(0.008) | 94(0.002) |
| **MNIST** [34] | 784 | 72000 | 10 | $\alpha = 0.01, b = 100, d = 8, hln = 100,$ $\lambda = 0.001, n_{neigh} = 2, \rho = 1$ | 92(0.1) | 94(0.2) | 95(1) | 97(0.6) |
| **NotMnist** [35] | 784 | 81000 | 10 | $\alpha = 0.01, b = 100, d = 9, hln = 100$ $\lambda = 0.001, n_{neigh} = 2, \rho = 1$ | 90(1) | 82(1.8) | 88(2) | 92(0.8) |
| **CIFAR10** [36] | 3072 | 50000 | 10 | $\alpha = 0.01, b = 100, d = 10, hln = 100$ $\lambda = 0.001, n_{neigh} = 2, \rho = 1$ | 81(7.21) | 80(8.00) | 82(12.11) | 84(8.77) |
| **Arcene** [37] | 10000 | 100 | 2 | $\alpha = 0.01, b = 100, d = 9, hln = 100, \lambda = 0.001$ $\lambda = 0.001, n_{neigh} = 2, \rho = 1$ | 69(0) | 61(0) | 51(0) | 78(0) |
| **Dexter** [37] | 20000 | 300 | 2 | $\alpha = 0.01, b = 100, d = 8, hln = 100$ $\lambda = 0.001, n_{neigh} = 2, \rho = 1$ | 71(0.7) | 77(0.9) | 81(1.1) | 80(1) |
| **Gisette** [37] | 5000 | 6000 | 2 | $\alpha = 0.01, b = 100, d = 8, hln = 100,$ $\lambda = 0.001, n_{neigh} = 2, \rho = 1$ | 92(0.25) | 93(0.21) | 91(0.3) | 98(0.07) |

**4.2.2. MNIST Data-set.** To start with, we will compare the proposed learning approach with and without the Zs for a varying number of hidden layers in the DNN. For this experiment, tanh activation functions are used. The learning rates $\eta_1$ for weights are set at 0.0001 whereas batch size is chosen as 128. The number of hidden layer neurons and the learning rate $\eta_2$ is chosen as 150 and 0.1 respectively. Zs are updated at each step for 100 times. The lambdas are kept fixed for this experimentation and the results are recorded in Table 2a. With an increase in the number of hidden layers in the DNN, the gap in the accuracy's achieved by the two approaches increase. Also, it is observed that the model with Zs performs significantly better than its counterpart.

Next, a seven hidden layers DNN model is trained for hundred iterations and the results are illustrated in Table 2a. The training accuracy for the case without the Zs does not improve beyond 15 epochs. On the other hand, the model with Z achieves an accuracy of 90% in less than 25 iterations. For the case without the Zs, this behavior is seen due to vanishing gradient [5] in deep neural networks. The problem does not occur in the model where Zs are used because the weights at each layer learn locally. To empirically



(a) (b) (c)

Figure 2. (a) Accuracy's of both models for 10 different sized hidden layers with tanh activation, (b) Progression of accuracy's of both models for 7 hidden layers, (c) Progression of the Total Cost for the two models

demonstrate this, the progression of the cost function with respect to the number of training iterations is shown in Figure 2c. The case with the Zs converges faster relative to the case without the Zs.

Next, the behavior of the methodology with respect to different parameters is studied. These results are shown in Table 3, where one parameter is changed while the others are kept fixed. Table 3a shows the results for different activation functions. Table 3b shows the comparison for different $\lambda$ values. Table 3c illustrates the impact of varying $\eta_1$ while Table 3d illustrates the impact of $\eta_2$.

It is observed from Table 3c, that when the value of $\eta_1$ is chosen as 0.1, the learning is poor because the weights converge faster than the corresponding Zs. This creates a peculiar problem, since Zs denote the data at output of every layer, the weights learn to predict based on the values of Z. However, if Zs do not reflect the data before the weights learn, the weights tend to learn from wrong representation of data thus leading to poor performance.

From our analysis, it was observed that the best case results were obtained when the DNN is chosen with four hidden layers containing fifty hidden neurons each. The batch size is chosen as 128 with tanh activation function. The regularization term $\alpha$ was fixed at 0.00001, whereas the penalty $\gamma$ is chosen at 10. The learning rates $\eta_1$ for the weights and $\eta_2$

were fixed at 0.0001 and 0.1 respectively. For each epoch, Zs were updated 100 times. The best case test accuracy achieved for the Mnist data-set at this parameter setting was 94.32%. Therefore, these parameters are used from here on for analysis on the Mnist data-set.

Next, the performance is analyzed on the problem of classification

Table 5. Parameters of the different models used in this paper.

| Architecture | Parameters |
|---|---|
| CNN | $n_{conv} = 2$, $n_{layers} = 2$, filter size = 5, pool shape = (3,3), hln= 128, $\alpha = 0.01$, $b = 100$, $\lambda = 0.001$ $n_{neigh} = 2$, $\rho = 1$ |
| MLNN | $d = 7$, hln= 128, $\alpha = 0.01$, $b = 100$, $\lambda = 0.001$ $n_{neigh} = 2$, $\rho = 1$ |
| SAE | $n_{units} = 3$, hln= 128, $\alpha = 0.01$, $b = 100$, $\lambda = 0.001$ $n_{neigh} = 2$, $\rho = 1$ |
| DAE | $n_{units} = 3$, hln= 128, $\alpha = 0.01$, $b = 100$, $\lambda = 0.001$ $n_{neigh} = 2$, $\rho = 1$ |

## 4.3. CLASSIFICATION

Table 6. % accuracies for the various data-sets with the proposed framework and four different archoitectures with generalization error in the parenthesis.

| Datasets | $p$ | $n$ | $\mathcal{F}$ | **MLNN** | **CNN** | **SAE** | **DAE** |
|---|---|---|---|---|---|---|---|
| **Rolling** | 11 | 35000 | 4 | 99(0.0001) | 99(0.0001) | 99(0.6) | 99(0.0) |
| **Sensorless** | 48 | 78000 | 11 | 94(0.008) | 94(0.008) | 95(0.008) | 94(0.002) |
| **MNIST** | 784 | 72000 | 10 | 94(0.2) | 95(1) | 97(0.6) | 97(0.6) |
| **NotMnist** | 784 | 81000 | 10 | 88(2) | 92(0.8) | 97(0.6) | 97(0.6) |
| **CIFAR10** | 3072 | 50000 | 10 | 81(7.21) | 80(8.00) | 82(12.11) | 84(8.77) |
| **Arcene** | 10000 | 100 | 2 | 69(0) | 61(0) | 51(0) | 78(0) |
| **Dexter** | 20000 | 300 | 2 | 71(0.7) | 77(0.9) | 81(1.1) | 80(1) |
| **Gisette** | 5000 | 6000 | 2 | 92(0.25) | 93(0.21) | 91(0.3) | 98(0.07) |

Next, we analyzed the performance of the proposed model on the data-sets mentioned in Table 6 and these results are presented in Table 6. To demonstrate the ability of the methodology to work with different architectures, we test CNN, DAE, SAE and MLNN where the parameters are summarized in 5.

The ability of the model to handle large complex problems by breaking it into simpler sub-problems and thus achieving faster learning is well illustrated by the testing accuracy results for the large data-sets such as Gas, Rolling and mnist, where the model has a very high testing accuracy (more than 90%). Since the data-sets Gas and Rolling are larger than the mnist data-set, hence for faster learning of the large data, the value of $\boldsymbol{\eta_2}$ for these two data-sets was set higher than that for mnist data-set.

Next, Four learning paradigms, namely DFA (Direct Feedback Alignment) [12], EDL (Error Driven Learning) [15], FA (Feedback Alignment) [38] and EDL (Error-driven Learning) [15], are tested with a total of ten data-sets. For all the learning paradigms, a multilayer NN with 10 hidden lauers is utilized. The results along with the model parameters are given in Table 4. Consistent performance is observed across data-sets with the proposed methodology and the results on large dimensional data-sets is very good too as observed from the results.

## 5. CONCLUSION

In this paper, sparsity is estimated as par of the learning problem in a structured process and efficient learning is observed when high dimensional data-sets are utilized. The proposed optimization approach achieves similar results with or without the additional variables. However, the use of additional variables improves performance in deep architectures. Parallelization and scalability is observed because a mathematical construct is used to break an optimization problem. A unique algorithm is used to demonstrate parallelization among the layers in the DNN as well as data-batches.

Overall, computational and analytical benefits are observed with the proposed methodology. Furthermore, the issue of incidental endogeneity will be addressed in the future.

## ACKNOWLEDGMENT

This research was supported in part by NSF I/UCRC award IIP 1134721 and Intelligent Systems Center.

## APPENDIX

**0.0.0.1. Proof:.** [Proof of Lemma 7] We rewrite the final three terms from the cost function in Eq. (6) as

$$\begin{aligned} H_1 = & \quad E[\boldsymbol{y}^T log(\hat{\boldsymbol{y}}(\boldsymbol{z}^{(d)}))^{-1}] + \sum_{i=1}^{d} \lambda_i P(\hat{\boldsymbol{W}}^{(i)}) \\ + & \sum_{i=2}^{d} \gamma_i log(1 - \frac{(\boldsymbol{z}^{(i)} - \hat{\boldsymbol{W}}^{(i)} f^{(i-1)}(\boldsymbol{z}^{(i-1)}))}{\gamma_i})^{-1} \end{aligned}$$

which can be rewritten as

$$\begin{aligned} H_1 = -E[\boldsymbol{y}^T log(\hat{\boldsymbol{y}}(\boldsymbol{z}^{(d)}))] + \sum_{i=1}^{d} \lambda_i |\hat{\boldsymbol{W}}^{(i)}| \\ - \sum_{i=2}^{d} \gamma_i log(1 - \frac{(\boldsymbol{z}^{(i)} - \hat{\boldsymbol{W}}^{(i)} f^{(i-1)}(\boldsymbol{z}^{(i-1)}))}{\gamma_i})^{-1} \end{aligned}$$

Observe that the first term in the cost function is the negative log of the error with respect to the variable $\boldsymbol{z}^{(d)}$ and the weight $\hat{\boldsymbol{W}}^{(d)}$. Furthermore, Let $\hat{\boldsymbol{W}}^{(i)}, \forall i$ not be dependent on the original function due to a functional nesting. The second term is also a log function depending for each layer $i$ is dependent on $\hat{\boldsymbol{W}}^{(i)}$ as $L_2$. Let $\hat{\boldsymbol{W}}^{(i)}$ depend on the log function without any nesting. Although, log is a concave function , negative log function is convex since negative of a concave function is convex [29]. However, the covexity only holds if there is no functional nesting within the variables. With these observations, it is clear that negative log function is inherently convex and is dependent on variables that have no functional nesting. Therefore, the first two terms are convex with respect to $\hat{\boldsymbol{W}}^{(d)}$.

Furthermore, since $|\hat{\boldsymbol{W}}^{(i)}|$ is a convex function with respect to the weights, we have a sum of three terms that are convex with respect to the weights of the DNN. Moreover, the convexity is also implied with respect to the additional variables.

Next, consider the case when both $d$ and the number of neurons in each layers tends to infinity and the formulation is interpreted as

$$\begin{aligned} H_1 = \lim_{d\to\infty} E[\boldsymbol{y}^T log(\hat{\boldsymbol{y}}(\boldsymbol{z}^{(d)}))^{-1}] + \sum_{i=1}^{d} \lambda_i |\hat{\boldsymbol{W}}^{(i)}| \\ + \sum_{i=2}^{d} \gamma_i log(1 - \frac{(\boldsymbol{z}^{(i)} - \hat{\boldsymbol{W}}^{(i)} f^{(i-1)}(\boldsymbol{z}^{(i-1)}))}{\gamma_i})^{-1} \end{aligned}$$

Let $\sum_{i=2}^{d} \gamma_i = L$ and let the convexity of log functions extends to large dimensions [30]. With these observations, it follows that $H_1$ for all the layers in the neural networks are linear combination of convex function that are independently convex [39] with respect to the parameters of the neural network.

Next, consider the fourth term in the cost function as

$$H_2 = -\Gamma^2 ||\hat{\boldsymbol{\lambda}}||^2. \tag{6}$$

Since norm is a convex function and the convexity extends to the case when the size of $||\hat{\boldsymbol{\lambda}}|| \to \infty$, it follows following the principles from earlier, $H^2$ is a concave function as $-H^2$ is convex.

Note that the overall cost function is given as $H_1 + H_2$. Observe that $H_1$ is only dependent on the parameters of the neural network and $H_0$ depends purely on the penalty parameters. The second derivative of the overall cost function with respect to $\hat{\boldsymbol{\theta}}$ reveals a convex cost function because the term due to the penalty parameters would result in a zero. Similarly, the second derivative of the overall cost function with respect to $\hat{\boldsymbol{\lambda}}$ reveals a concave cost function. It follows that the overall cost function is convex relative to the parameters of the neural network while it is concave with respect to the penalty parameters. □

**0.0.0.2. Proof:.** [Proof of Theorem 4] The proof follows directly as a consequence of Lemma 7. The logic follows from the fact that $\hat{\boldsymbol{\lambda}}$ is a maximizer for the cost function [40] due to concavity of the cost function. Similarly, it follows that $\boldsymbol{\theta}^*$ is a minimizer for the cost function due to convexity with respect to $\hat{\boldsymbol{\theta}}$. Thus, invoking the minimax theorem in [41] it can be seen that $(\boldsymbol{\lambda}^*, \boldsymbol{\theta}^*)$ is a saddle point solution to the game and the game admits a Nash equilibrium. □

## REFERENCES

[1] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[2] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio, "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization," in *Advances in neural information processing systems*, 2014, pp. 2933–2941.

[3] M. Jaderberg, W. M. Czarnecki, S. Osindero, O. Vinyals, A. Graves, and K. Kavukcuoglu, "Decoupled neural interfaces using synthetic gradients," *arXiv preprint arXiv:1608.05343*, 2016.

[4] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[5] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning*, 2013, pp. 1310–1318.

[6] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[7] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, "The loss surfaces of multilayer networks," in *Artificial Intelligence and Statistics*, 2015, pp. 192–204.

[8] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[9] H. Mobahi and J. W. Fisher III, “A theoretical analysis of optimization by gaussian continuation.” in *AAAI*, 2015, pp. 1205–1211.

[10] L. Vese, “A method to convexify functions via curve evolution,” *Communications in Partial Differential Equations*, vol. 24, no. 9-10, pp. 1573–1591, 1999.

[11] D.-H. Lee, S. Zhang, A. Fischer, and Y. Bengio, “Difference target propagation,” in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2015, pp. 498–515.

[12] A. Nøkland, “Direct feedback alignment provides learning in deep neural networks,” in *Advances in Neural Information Processing Systems*, 2016, pp. 1037–1045.

[13] G. Taylor, R. Burmeister, Z. Xu, B. Singh, A. Patel, and T. Goldstein, “Training neural networks without gradients: A scalable admm approach,” in *International Conference on Machine Learning*, 2016, pp. 2722–2731.

[14] M. Carreira-Perpinan and W. Wang, “Distributed optimization of deeply nested systems,” in *Artificial Intelligence and Statistics*, 2014, pp. 10–19.

[15] R. Krishnan, S. Jagannathan, and V. Samaranayake, “Direct error driven learning for deep neural networks with applications to big data,” *Procedia Computer Science*, vol. 144, pp. 89–95, 2018.

[16] J. Fan, F. Han, and H. Liu, “Challenges of big data analysis,” *National science review*, vol. 1, no. 2, pp. 293–314, 2014.

[17] G. Kang, J. Li, and D. Tao, “Shakeout: A new regularized deep neural network training scheme.” in *AAAI*, 2016, pp. 1751–1757.

[18] H. Xu, C. Caramanis, and S. Mannor, “Sparse algorithms are not stable: A no-free-lunch theorem,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 1, pp. 187–193, 2012.

[19] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[20] J. Keuper and F.-J. Preundt, “Distributed training of deep neural networks: theoretical and practical limits of parallel scalability,” in *Proceedings of the Workshop on Machine Learning in High Performance Computing Environments*. IEEE Press, 2016, pp. 19–26.

[21] O. Gupta and R. Raskar, “Distributed learning of deep neural network over multiple agents,” *Journal of Network and Computer Applications*, vol. 116, pp. 1–8, 2018.

[22] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le *et al.*, "Large scale distributed deep networks," in *Advances in neural information processing systems*, 2012, pp. 1223–1231.

[23] S. Zhang, A. E. Choromanska, and Y. LeCun, "Deep learning with elastic averaging sgd," in *Advances in Neural Information Processing Systems*, 2015, pp. 685–693.

[24] M. Hardt, "Train Faster, Generalize Better: Stability of Stochastic Gradient Descent," *Mathematical and Computational Foundations of Learning Theory*, p. 64, 2015.

[25] S. Sonoda and N. Murata, "Neural network with unbounded activation functions is universal approximator," *Applied and Computational Harmonic Analysis*, 5 2015.

[26] P. Niyogi and F. Girosi, "On the relationship between generalization error, hypothesis complexity, and sample complexity for radial basis functions," *Neural Computation*, vol. 8, no. 4, pp. 819–842, 1996.

[27] N. T. Trendafilov and I. T. Jolliffe, "Dalass: Variable selection in discriminant analysis via the lasso," *Computational Statistics & Data Analysis*, vol. 51, no. 8, pp. 3718–3736, 2007.

[28] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005.

[29] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[30] B. Klartag, "High-dimensional distributions with convexity properties," *Preprint, http://www. math. tau. ac. il/klartagb/papers/euro. pdf*, 2010.

[31] R. Isaacs, "Differential games ii," 1954.

[32] A. Soylemezoglu, S. Jagannathan, and C. Saygin, "Mahalanobis taguchi system (mts) as a prognostics tool for rolling element bearing failures," *Journal of Manufacturing Science and Engineering*, vol. 132, no. 5, p. 051014, 2010.

[33] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available:

[34] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[35] Y. Bulatov, "Notmnist dataset," *Google (Books/OCR), Tech. Rep.[Online].* 2011.

[36] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[37] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror, “Result analysis of the nips 2003 feature selection challenge,” in *Advances in neural information processing systems*, 2005, pp. 545–552.

[38] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman, “Random synaptic feedback weights support error backpropagation for deep learning,” *Nature communications*, vol. 7, p. 13276, 2016.

[39] L. Vandenberghe and S. Boyd, “Semidefinite programming,” *SIAM review*, vol. 38, no. 1, pp. 49–95, 1996.

[40] S. Wright and J. Nocedal, “Numerical optimization,” *Springer Science*, vol. 35, no. 67-68, p. 7, 1999.

[41] M. Sion *et al.*, “On general minimax theorems.” *Pacific Journal of mathematics*, vol. 8, no. 1, pp. 171–176, 1958.

**SECTION**

## 2. CONCLUSIONS AND FUTURE WORK

In summary, a two-step approach to classification is considered in this dissertation where the first step is dimension reduction and the second step is classification. Each step is designed in such a way that the main goal is to address the challenges introduced by the high dimensionality and volume of big-data. In the first two papers, novel multi-step dimension-reduction approaches are introduced where it is demonstrated that the challenges such as noise, spurious correlation and heterogeneity are well addressed.

In the second part of the dissertation we focus on deep neural network-based classification. Here, we introduce novel learning framework where challenges such as generalization error, vanishing gradients and unsatisfactory solutions due to non-convexity of optimization problem are addressed. The efficiency of the overall two-step framework for big-data analysis is demonstrated both theoretically and in simulation. The conclusions for each paper is presented next.

### 2.1. CONCLUSIONS

In the first paper, a generalized distance measure with a novel hierarchical dimension-reduction approach (GDM-HDR) was introduced. GDM-HDR appears to outperform Mahalanobis distance (MD) when noisy/redundant dimensions are present in the data. Due to the use of L1 norm in GDM-HDR, distance concentration was not observed. Computational complexity for GDM-HDR was linear which appears to be an improvement over the cubic complexity observed in MD. Because dependencies are calculated in groups, HDR is able to satisfy the rank condition even when the number of dimensions in the data far exceeds the number of data-points thus allowing efficient dimension reduction. HDR is able to capture more than 90% of the information from the data. Overall, HDR appears to outperform standard dimension-reduction approaches in large dimensional scenarios. HDR fails to perform properly, when nonlinear relationships are present in the data.

In the second paper, a multi-step dimension-reduction approach (NDR) was proposed to address nonlinear relationships, heterogeneity and noisy dimensions. Because an information-loss criterion was introduced as a user-defined parameter in NDR, the proposed methodology was able to dynamically determine the number of dimensions to be extracted from the data in contrast with standard dimension reduction procedures and HDR. Novel aggregation methods allowed the methodology to process data in batches while allowing complete exploration of the data distribution. Due to the use of distance covariance, NDR was able to capture nonlinear relationships from the data while exhibiting 24% improvement over PCA for the sensorless drive diagnostics data-set. Since, NDR and HDR do not obtain feedback from the detection methodology both suffer from ineffectiveness in very complicated data-sets, such as CiFAR-10.

In the third paper, a classifier design is presented where synthetic distortions are utilized to approximate the impact of noisy data. Due to this, the proposed approach was able to better compensate for data distortions relative to SGD. Careful selection of the magnitude of the perturbation is necessary because large perturbations deteriorate the performance of the proposed framework significantly. The direct use of the overall error signal in the learning process appears to mitigate the vanishing gradient issue even when sigmoid and tanh activation functions were used. The proposed framework combined with the proposed learning regime appears to outperform other approaches such as DFA and FA in all of the data-sets under analysis and a 8% improvement is observed in simulation.

In the fourth paper, a two player game was designed to alleviate the challenge from the third paper where the exact measure of perturbations need not be premeditated. A generic distortion model was designed to introduce distortions in the learning problem which increases the applicability of the proposed approach. A two player game was developed where the optimal compensation for the classifier and the corresponding optimal distortion was obtained as part of the solution. Due to this the proposed approach is more resilient to distribution changes. Overall, the proposed approach appears to provide a 5% improvement in generalization error and a 10% improvement in accuracy over SGD in the presence of domain shift.

Finally, in the last paper, sparsity is estimated as part of the learning problem in a structured process and efficient learning is observed when high dimensional data-sets are utilized. The proposed $L_1$ regularized optimization approach achieves similar results with or without the additional variables which supports our theoretical results. However, the use of additional variables improves performance in deep architectures where a 15% improvement is demonstrated over gradient descent-based approaches during the training process of a ten layer neural network. Parallelization and scalability is observed because a mathematical construct is used to break an optimization problem where parallelization among the layers in the DNN as well as data-batches is demonstrated. A 200% computational improvement is demonstrated.

Overall, computational and analytical benefits are observed with the proposed suite of methodologies for big-data in the presence of big-data challenges. Some additional perspectives on the dissertation are introduced next.

## 2.2. FUTURE WORK

The requirement of an off-line learning phase stems from the assumption that the data-distribution in the learning phase and the testing phase are similar. However, this assumption is not realistic and typically, in practical big-data scenarios non-stationarity of data-distribution is observed. To incorporate this phenomenon in the classifier, a combination of online and offline framework must be designed where the learning problem must be redefined to include information about distribution changes. Since these distribution changes are typically large and fast, this is a very challenging problem.

This dissertation could also be extended to simultaneously analyze different types of data-set such as speech, images, numeric, etc, where the idea is to make a decision taking into account observations provided by different data-types. To enable such an analysis, one would have to design a basis transformation, such as wavelet transformation, that converts all of these data-types into one common domain and then learn from the common domain. Since, all of these observations are representative of the task at hand, one can design a generative model to learn the probability distribution of the task itself which could then be used for decision-making.

One the other hand, this dissertation was focused on the application of classification and the problem of regression/prediction was largely ignored. Therefore, one can consider the regression and address big-data challenges such as incidental endogeneity and data-speed which is typically in giga-bytes per second. To handle such problems, one would start with a stationary and stochastic process where the future values must be predicted based on the history of the process and then redesign the learning problem to address big-data challenges. Applications of such an approach would include natural language processing, fault prognostics, etc.

Finally, throughout the dissertation, we have ignored application specific domain knowledge in the design of our models. In the future, we can consider applications such as cyber physical systems, neuroscience, genetics, etc and extend the techniques presented in this dissertation to incorporate such domain expertise. Since this is a very active field of research, the scope of such applications is enormous.

## REFERENCES

[1] "What is Big Data?" Big Data, 8 Nov. 2017, bigdata.black/featured/what-is-big-data/.

[2] J. Fan, F. Han, and H. Liu, “Challenges of big data analysis,” *National science review*, vol. 1, no. 2, pp. 293–314, 2014.

[3] Wu, Xindong, Xingquan Zhu, Gong-Qing Wu, and Wei Ding. "Data mining with big data." IEEE transactions on knowledge and data engineering 26, no. 1 (2014): 97-107.

[4] Zhong, Guoqiang, Li-Na Wang, Xiao Ling, and Junyu Dong. "An overview on data representation learning: From traditional feature learning to recent deep learning." The Journal of Finance and Data Science 2, no. 4 (2016): 265-278.

[5] Thusoo, Ashish, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Suresh Anthony, Hao Liu, Pete Wyckoff, and Raghotham Murthy. "Hive: a warehousing solution over a map-reduce framework." Proceedings of the VLDB Endowment 2, no. 2 (2009): 1626-1629.

[6] Hadoop, Apache. "Apache hadoop." (2011).

[7] C. Giraud, *Introduction to high-dimensional statistics*. CRC Press, 2014, vol. 138.

[8] Fan, Jianqing, and Yuan Liao. "Endogeneity in high dimensions." Annals of statistics 42, no. 3 (2014): 872.

[9] C. P. Chen and C.-Y. Zhang, “Data-intensive applications, challenges, techniques and technologies: A survey on big data,” *Information Sciences*, vol. 275, pp. 314–347, 2014.

[10] Xu, Yan, Yanming Sun, Jiafu Wan, Xiaolong Liu, and Zhiting Song. "Industrial Big Data for Fault Diagnosis: Taxonomy, Review, and Applications." IEEE Access (2017).

[11] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2002.

[12] R. A. Johnson and D. W. Wichern, *Applied multivariate statistical analysis*. Prentice hall Englewood Cliffs, NJ, 1992, vol. 4.

[13] M. Balasubramanian and E. L. Schwartz, "The isomap algorithm and topological stability," *Science*, vol. 295, no. 5552, pp. 7–7, 2002.

[14] S. T. Roweis and L. K. Saul, “Nonlinear dimensionality reduction by locally linear embedding,” *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[15] D. L. Donoho and C. Grimes, “Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data,” *Proceedings of the National Academy of Sciences*, vol. 100, no. 10, pp. 5591–5596, 2003.

[16] M. Belkin and P. Niyogi, “Laplacian eigenmaps for dimensionality reduction and data representation,” *Neural computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[17] B. Schölkopf, A. Smola, and K.-R. Müller, “Kernel principal component analysis,” in *International Conference on Artificial Neural Networks*. Springer, 1997, pp. 583–588.

[18] K. P. Adragni, E. Al-Najjar, S. Martin, S. K. Popuri, and A. M. Raim, “Group-wise sufficient dimension reduction with principal fitted components,” *Computational Statistics*, vol. 31, no. 3, pp. 923–941, 2016.

[19] Z. Guo, L. Li, W. Lu, and B. Li, “Groupwise dimension reduction via envelope method,” *Journal of the American Statistical Association*, vol. 110, no. 512, pp. 1515–1527, 2015.

[20] A. D. Ward and G. Hamarneh, “The groupwise medial axis transform for fuzzy skeletonization and pruning,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 32, no. 6, pp. 1084–1096, 2010.

[21] J. Zhou, J. Wu, and L. Zhu, “Overlapped groupwise dimension reduction,” *Science China Mathematics*, vol. 59, no. 12, pp. 2543–2560, 2016.

[22] A. Vehtari, A. Gelman, and J. Gabry, “Practical bayesian model evaluation using leave-one-out cross-validation and waic,” *Statistics and Computing*, vol. 27, no. 5, pp. 1413–1432, 2017.

[23] P. Marttinen, S. Myllykangas, and J. Corander, “Bayesian clustering and feature selection for cancer tissue samples,” *BMC bioinformatics*, vol. 10, no. 1, p. 90, 2009.

[24] R. S. Sutton, S. D. Whitehead *et al.*, “Online learning with random representations,” in *Proceedings of the Tenth International Conference on Machine Learning*, 1993, pp. 314–321.

[25] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[26] H. W. Lin and M. Tegmark, “Why does deep and cheap learning work so well?” *arXiv preprint arXiv:1608.08225*, 2016.

[27] M. Hardt, "Train Faster, Generalize Better: Stability of stochastic gradient descent,” *Mathematical and Computational Foundations of Learning Theory*, p. 64, 2015.

[28] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training recurrent neural networks,” in *International Conference on Machine Learning*, 2013, pp. 1310–1318.

[29] M. S. Advani and A. M. Saxe, “High-dimensional dynamics of generalization error in neural networks,” *arXiv preprint arXiv:1710.03667*, 2017.

[30] Z. Yu, L. Li, J. Liu, and G. Han, “Hybrid adaptive classifier ensemble,” *IEEE Transactions on Cybernetics*, vol. 45, no. 2, pp. 177–190, Feb 2015.

[31] J. Goldberger and E. Ben-Reuven, “Training deep neural-networks using a noise adaptation layer,” 2016.

[32] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich, “Training deep neural networks on noisy labels with bootstrapping,” *arXiv preprint arXiv:1412.6596*, 2014.

[33] Srivastava, Nitish, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting." Journal of machine learning research 15, no. 1 (2014): 1929-1958.

[34] Miyato, Takeru, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. "Virtual Adversarial Training: a Regularization Method for Supervised and Semi-supervised Learning." arXiv preprint arXiv:1704.03976 (2017).

[35] D. Mishkin and J. Matas, “All you need is a good init,” *arXiv preprint arXiv:1511.06422*, 2015.

[36] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, “Understanding deep learning requires rethinking generalization,” *arXiv preprint arXiv:1611.03530*, 2016.

[37] T. Xiao, H. Li, W. Ouyang, and X. Wang, “Learning deep feature representations with domain guided dropout for person re-identification,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1249–1258.

[38] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting.” *Journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[39] J. Lemley, S. Bazrafkan, and P. Corcoran, “Smart augmentation learning an optimal data augmentation strategy,” *IEEE Access*, vol. 5, pp. 5858–5869, 2017.

[40] Y. Taigman, A. Polyak, and L. Wolf, “Unsupervised cross-domain image generation,” *arXiv preprint arXiv:1611.02200*, 2016.

[41] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, “Domain-adversarial training of neural networks,” *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[42] M. Jaderberg, W. M. Czarnecki, S. Osindero, O. Vinyals, A. Graves, and K. Kavukcuoglu, “Decoupled neural interfaces using synthetic gradients,” *arXiv preprint arXiv:1608.05343*, 2016.

[43] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural Networks*, vol. 61, pp. 85–117, 2015.

[44] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, “The loss surfaces of multilayer networks,” in *Artificial Intelligence and Statistics*, 2015, pp. 192–204.

[45] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[46] D.-H. Lee, S. Zhang, A. Fischer, and Y. Bengio, “Difference target propagation,” in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2015, pp. 498–515.

[47] A. Nøkland, “Direct feedback alignment provides learning in deep neural networks,” in *Advances in Neural Information Processing Systems*, 2016, pp. 1037–1045.

# VITA

Krishnan Raghavan was born in a small town named Kumbakonam, Tamilnadu, India in the year 1990. He received his primary education in Rajasthan, India. In July of 2008, He moved to Mumbai, Maharashtra for his bachelors degree. In 2012, he received his Bachelors degree in Instrumentation Engineering from Vivekanand Education Society Institute of Technology that is affiliated with University of Mumbai.

After receiving his bachelors degree, he moved to U.S.A., where he joined the Missouri University of Science and Technology, Rolla as a graduate student in the Electrical and Computer Engineering Department to pursue a Master of Science degree with the thesis option. He received his Masters Degree in Computer Engineering in the December 2014. He then started his doctorate under the guidance of Dr. S. Jagannathan and Dr. V.A. Samaranayake. In May of 2019, he received his Ph.D. in Computer Engineering from Missouri University of Science and Technology.