DCA-based machine learning techniques with applications in finance and healthcare

van Tuan Pham

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Techniques d'apprentissage automatique basées sur DCA avec des applications dans la finance et la santé

THÈSE

présentée et soutenue publiquement le 24 novembre 2023

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

PHAM Van Tuan

Composition du jury

Président :	Yann Guermeur	Directeur de recherche, CNRS
Rapporteurs :	Mustapha Lebbah	Professeur, Université Paris-Saclay
	Adnan Yassine	Professeur, Université Le Havre Normandie
Examinateur :	Hanene Azzag	MCF-HDR, LIPN Université Paris 13
Directrice de thèse :	Hoai An LE THI	Professeur, Université de Lorraine
Co-directeur de thèse :	Pascal Damel	MCF-HDR, Université de Lorraine

Laboratoire de Génie Informatique, de Production et de Maintenance — EA 3096

Mis en page avec la classe thesul.

Remerciements

Tout d'abord, je tiens à exprimer ma sincère gratitude à ma directrice de thèse, la Professeure LE THI Hoai An à LGIPM/IA - Université de Lorraine. Elle m'a non seulement offert l'opportunité de venir en France pour la première fois en tant que stagiaire de Master, mais a également joué un rôle déterminant dans le lancement de ma carrière de recherche scientifique. Ses conseils, sa patience, son dévouement et la richesse de connaissances qu'elle a partagées ont été cruciaux pour me permettre de compléter ma thèse du mieux que je le pouvais.

Je tiens sincèrement à exprimer ma profonde gratitude spéciale envers mon co-directeur de thèse, le MCF-HDR Pascal DAMEL, pour ses commentaires utiles et constructifs, ses conseils et son soutien inébranlable.

Je tiens à exprimer ma gratitude envers le Professeur Mustapha Lebbah de l'Université Paris-Saclay et le Professeur Adnan Yassine de l'Université Le Havre Normandie pour avoir accepté d'être les rapporteurs de ma thèse. J'apprécie le temps précieux que vous avez consacré à la lecture de ma thèse.

Je souhaite exprimer ma gratitude profonde envers le Professeur Yann Guermeur du CNRS et MCF-HDR Hanene Azzag de l'Université Paris 13 pour avoir accepté d'être membres du comité de ma thèse.

Je suis profondément reconnaissant au Professeur Ahmed Zidna de l'Université de Lorraine et au Professeur François Charoy de l'Université de Lorraine pour le temps qu'ils ont consacré et leur soutien en tant que membres de mon comité de suivi.

Ma gratitude particulière est adressée au Professeur PHAM Dinh Tao de l'INSA Rouen pour son encouragement inestimable et les précieuses références qu'il a fournies dans mon travail de recherche.

Je souhaite également remercier chaleureusement le Dr. Luu Hoang Phuc Hau pour son généreux soutien et nos discussions enrichissantes, en particulier dans les domaines de l'optimisation et de l'apprentissage automatique tout au long de ma recherche. Il a toujours fait preuve d'une volonté de répondre à mes questions de manière approfondie et attentive.

Je suis profondément reconnaissant envers le Professeur Le Hoai Minh et le Dr. Ho Vinh Thanh pour leur aide précieuse et nos discussions approfondies tout au long de notre collaboration.

J'aimerais exprimer ma reconnaissance envers mes collègues à LGIPM/IA et mes amis, notamment Phuc Hau, Tuyet Trinh, My Le, Duc Vinh, Khac Linh, Manh Cuong, Duong Trinh, Moez, Mlle. Rose, UV, Truc Vy, Hoang Viet et les autres qui m'ont apporté leur soutien, leur encouragement et ont partagé de merveilleux moments au cours de nos trois années ensem-

ble. Un merci spécial à Mai Chan, Van Giang, TRAN Bach et The Anh pour leurs conseils inestimables, tant dans la vie quotidienne que dans la recherche en France.

Je tiens également à exprimer ma gratitude envers Mme. Aurélie et Mme. Olivia, les secrétaires de LGIPM/IA, pour leur disponibilité exceptionnelle, leur enthousiasme et leur assistance dévouée.

Je souhaite adresser des remerciements particuliers à ma femme NGUYEN Thi Huong, qui a toujours été à mes côtés, m'encourageant et me soutenant, m'aidant à surmonter les défis et les difficultés. Merci à ma fille, PHAM Ngoc Gia An, qui est un enfant bien élevé et aimant, et qui a été une grande source de motivation pour moi tout au long du processus de réalisation de cette thèse. J'envoie toute mon affection à mes parents et à tous les membres de ma famille.

Je dédie cette thèse à ma famille,
mes parents,
ma femme et ma fille,
qui m'ont soutenu tout au long de ce parcours.

PHAM Van Tuan

Né le 4 avril 1989 au Vietnam

E-mail: van-tuan.pham@univ-lorraine.fr

Adresse professionnelle: Bureau UM-AN1-040, LGIPM - Université de Lorraine, 3 rue Augustin Fresnel, BP 45112, 57073 METZ Cedex 03, France

1. Situation Actuelle

Depuis 10/2020	Doctorant au LGIPM, Université de Lorraine, France, encadré par le Prof. Hoai An Le Thi et le Prof. Pascal Damel Sujet de thèse: **Techniques d'apprentissage automatique basées sur DCA avec applications en finance et en santé**
Depuis 08/2012	Enseignant-chercheur en Informatique économique à l'Université Nationale d'économie, Hanoi, Vietnam

2. Expérience Professionnelle

08/2019 - 12/2019	Ingénieur en données et Développeur web chez VinTech Group, Hanoi, Vietnam
04/2018 - 07/2019	Product owner en IA lab chez NAL Jsc., Hanoi, Vietnam
12/2017 - 03/2018	Ingénieur en apprentissage automatique chez NAL Jsc., Hanoi, Vietnam
10/2016 - 11/2017	Ingénieur en apprentissage automatique chez ChappieBot Jsc., Hanoi, Vietnam
04/2016 - 09/2016	Stagiaire au laboratoire LITA, UFR MIM, Université de Lorraine, Metz, France Mémoire: **Optimisation de contrôle par les techniques d'apprentissage automatique pour un Véhicule de Surface Autonome**

3. Diplômes et Formations

2020 - 2023	Doctorant en Informatique, LGIPM, Université de Lorraine, Metz, France
2015 - 2016	Master 2 en Technologies de l'information et de la communication, Université de Lorraine, Metz, France
2014 - 2015	Master 1 en Technologies de l'information et de la communication, Université des Sciences et de la Technologie de Hanoi (USTH), Hanoi, Vietnam
2008 - 2012	Diplôme Universitaire en Systèmes de gestion d'informations, Université nationale d'économie, Hanoi, Vietnam

4. Projet durant le doctorat

Du 10/2020 au 06/2021

Project: *"Robot navigation"*
Description: L'objectif du projet est de créer une méthode d'apprentissage pour la navigation automatique d'un robot terrestre. Notre méthode a été efficacement éprouvée sur un robot en situation réelle. Les détails du projet ne sont pas divulgués dans ce document en raison de leur caractère confidentiel.
Responsable : Prof.Hoai An Le Thi - **Partenariat industriel** : Naval Group
Role: Participer à des recherches sur une technique de navigation basée sur l'apprentissage supervisé, en utilisant les informations des capteurs du robot tels que : Lida, caméra, IMU sur des robots réels et des robots simulés sur environnement Gazebo/ROS.

Publications

Referred papers

[1] Pham, V.T., Luu, H.P.H., Le Thi, H.A. (2022). A Block Coordinate DCA Approach for Large-Scale Kernel SVM. In: Nguyen, N.T., Manolopoulos, Y., Chbeir, R., Kozierkiewicz, A., Trawiński, B. (eds) Computational Collective Intelligence. ICCCI 2022. Lecture Notes in Computer Science(), vol 13501. Springer, Cham. https://doi.org/10.1007/978-3-031-16014-1_27.

[2] Pham, V.T., Le Thi, H.A., Luu, H.P.H., Damel, P. (2023). DCA-Based Weighted Bagging: A New Ensemble Learning Approach. In: Nguyen, N.T., et al. Intelligent Information and Database Systems. ACIIDS 2023. Lecture Notes in Computer Science(), vol 13996. Springer, Singapore. https://doi.org/10.1007/978-981-99-5837-5_11

[3] Pham, V.T., Le Thi, H.A. and Damel, P. (2023). Cost-sensitive weighted bagging DCA based method for imbalanced financial data. **Submitted** in: *Proceedings of the 4th International Conference and Summer School on Numerical Computations: Theory and Algorithms NUMTA 2023*.

Contents

Introduction générale **5**

Chapter 1
Methodologies and Fundamentals **11**

1.1 Machine learning and challenges 11
1.1.1 Machine learning 12
1.1.2 Data's challenges 12
1.2 Prerequisite knowledge for our methods 15
1.2.1 Coordinate Descent method 15
1.2.2 Ensemble learning and Bagging 18
1.2.3 Deep learning and Transfer learning 21
1.3 DC Programming and DCA 24
1.3.1 Fundamental convex analysis 25
1.3.2 DC Programming 27
1.3.3 Standard DC Algorithm (DCA) 30

Chapter 2
A block coordinate DCA approach for large-scale kernel SVM

2.1 Introduction 33
2.1.1 Context and related work 33
2.1.2 Motivation and contributions 34
2.2 Optimization problem of kernel SVM 35
2.3 Block coordinate DCA based method 36
2.3.1 Block coordinate DCA algorithm 36
2.3.2 Application to kernel SVM with LS-DC losses 37
2.3.3 Block selection rule 38
2.3.4 Block coordinate DCA for kernel SVM algorithm 39
2.4 Datasets and numerical settings 39
2.4.1 Datasets 40

2.4.2 Set up experiments and parameters 41
2.5 Experiments and Discussion 41
2.5.1 Experiment 1: Performance on large benchmark datasets 41
2.5.2 Experiment 2: Parameter analysis 42
2.5.3 Experiment 3: Robustness on difference loss functions 43
2.6 Conclusion 44

Chapter 3
DCA-based bagging approach and application in financial imbalanced data

3.1 Introduction and related works 48
3.1.1 Ensemble learning and bagging 48
3.1.2 Financial problems 49
3.1.3 Our contribution 49
3.2 Weighted bagging approach 50
3.2.1 Ensemble learning and Bagging method 50
3.2.2 Weighted bagging 50
3.3 Optimization problem based on DCA 52
3.4 BaggingDCA performance on popular benchmark dataset 53
3.4.1 Weighted bagging DCA algorithm 53
3.4.2 BaggingDCA performance on popular benchmark dataset 54
3.4.3 Experimental setting and Datasets 55
3.4.4 Experiment 1: Evaluating BaggingDCA method on popular benchmark datasets 56
3.4.5 Experiment 2: Parameter analysis with different bagging sizes 58
3.5 Cost-sensitive Weighted Bagging DCA-based algorithm for imbalanced data . . 60
3.5.1 Cost-sensitive learning 60
3.5.2 Cost-sensitive weighted bagging algorithm 60
3.5.3 Experimental setting 61
3.5.4 Imbalanced financial datasets 63
3.5.5 Experiment 3: Data exploration and feature selection with the statistical test on Luxembourg Credit approval dataset 64
3.5.6 Experiment 4: Assessing the efficacy of the CSB-DCA method in handling imbalanced data across varied financial datasets 67
3.5.7 Experiment 5: Evaluating the robustness of the CSB-DCA method on imbalanced datasets 68
3.6 Conclusion 69

Chapter 4
Deep transfer learning with MCSDCA and applications in Healthcare problems

4.1 Introduction . . . 71
4.2 Background . . . 73
4.2.1 Deep transfer learning in NLP . . . 73
4.2.2 Pre-trained language models . . . 77
4.3 Our methods . . . 80
4.3.1 Markov Chain Stochastic DCA as DL optimizer . . . 80
4.3.2 Hybrid CNN - BiLSTM architecture . . . 81
4.3.3 Hybrid CNN-BiLSTM with Word2Vec embedding . . . 83
4.3.4 Hybrid CNN-BiLSTM with BERT embedding . . . 84
4.3.5 Hybrid CNN-BiLSTM with RoBERTa embedding . . . 85
4.4 Healthcare text classification problems and Datasets . . . 86
4.4.1 Hallmarks of Cancer (HoC) . . . 86
4.4.2 Extracting action items for physicians from hospital discharge notes (CLIP) . . . 87
4.4.3 Medical transcriptions (MedTrans) . . . 87
4.4.4 Datasets' information . . . 87
4.5 Experiment setting . . . 88
4.5.1 Comparative algorithms and optimizers . . . 88
4.5.2 Experiment setup and model setting . . . 88
4.5.3 Evaluation metrics . . . 89
4.6 Numerical experiment and Discussion . . . 90
4.6.1 Experiment 1: Comparison of CNN, LSTM and the proposed CNN-BiLSTM with Word2Vec . . . 90
4.6.2 Experiment 2: Comparison of pre-trained language models . . . 91
4.7 Conclusion . . . 91

Chapter 5
Conclusion and perspectives

Bibliography **95**

List of Figures

1.1 Illustrate the Coordinate Descent (CD) method applied to the function $f(x, y) = 3x^2 - 4xy + 5y^2$, demonstrating the step-by-step iterative updates along the x and y coordinates, leading to the approach of the minimum value. Starting from the point $(x, y) = (-1.5, -1)$ and first performing line-search along the x-axis, the CD algorithm alternates between updating the x coordinate and the y coordinate in each iteration. This iterative process continues until it reaches the minimum value at $f(0, 0) = 0$ (or stop criterion is met). 17

1.2 Illustration of ensemble learning (bagging). Each base model takes a subset of samples from the original dataset. The base models are then trained sequentially or in parallel to produce diverse predictions. The ensemble model combines the predictions of the base models, usually by taking a majority vote or averaging, to make the final prediction. 19

1.3 Intuitive about transfer learning. Taking inspiration from how people share knowledge between different areas, transfer learning seeks to use what we know from one related field to make learning better in a different area. 21

1.4 Comparision of the learning process between Traditional ML and Transfer learning. While Traditional ML constructs models from scratch using the available dataset and domain-specific features, Transfer learning aims to extract the knowledge from one or more source tasks and applies the knowledge to a target task. 22

1.5 Word2Vec architecture 23

1.6 Architecture of BERT_Base and BERT_Large (Extracted from HuggingFace. BERT_Base comprises 12 layers of Transformer encoders, a specialized neural network architecture, whereas BERT_Large incorporates 24 layers of Transformer encoders 24

2.1 Performances of BC-DCASVM on CIFA10-CP (2.1(a)) and NEWS20 (2.1(b)) as the number of epochs increases. 43

3.1 Performances of related algorithms on benchmark datasets as the number of base-learners increases. 59

3.2 F-test and T-test results. 65
3.3 Features correlation matrix of the Luxembourg credit approval dataset. Both the X and Y axes show different attributes present in the dataset. 66

4.1 Visualization of a neural network composed of an input layer, an output layer, and two hidden layers in between. 74
4.2 A convolutional neural network (CNN) architecture 75
4.3 LSTM standard architecture. 76
4.4 A convolutional neural network (CNN) acquires basic features in the first layers, progresses to complex intermediate layers, and results in sophisticated high-level feature representations that are then used for classification. 77
4.5 Word2Vec is capable of capturing multiple levels of similarity between words, allowing semantic and syntactic patterns to be reproduced using numerical vectors. For example, in the pairs of the words "man", "woman" and "king", and "queen", two of them refer to males and two to females. 78
4.6 The proposed CNN-BiLSTM model architecture diagram. 83
4.7 The proposed CNN-BiLSTM model architecture with Word2Vec embedding. . 84
4.8 Proposed CNN-BiLSTM model architecture with BERT embedding. 85
4.9 The proposed CNN-BiLSTM model architecture with RoBERTa embedding. . 86

List of Tables

2.1 Information of the datasets (samples (m), features (n)) used in our experiments. 40
2.2 Performances of the related algorithms on 10 benchmark datasets. All results are averaged over 5 folds with the standard deviations. 42
2.3 Performances of the two unified algorithms: BC-DCASVM and UniSVM with three loss functions: Truncated least square (Trunc-Least-Square), truncated squared Hinge (Trunc-Squared-Hinge) and Squared Hinge. 44

3.1 Nine benchmark datasets from LibSVM for evaluating the performance of BaggingDCA 56
3.2 Performance comparison (mean accuracy±std.) of related algorithms on nine benchmark datasets. 57
3.3 Datasets for evaluation (dataset, abbreviation(abbr.), feature(fea.), instance(ins.), positive(pos.), negative(neg.), class imbalance ratio(imb.ratio)). 64
3.4 Statistics descriptive of the Luxembourg credit approval dataset. 66
3.5 Performance of the four algorithms in handling imbalanced data across five financial datasets 67
3.6 The performance of four algorithms on five financial datasets after introducing five random features. 69

4.1 Information of the datasets (Number of the trainset, valset, testset and total number of samples in our experiments. 87
4.2 Results of the proposed CNN-BiLSTM architecture and traditional methods with Word2Vec embedding on the 3 healthcare datasets. 90
4.3 Result of CNN-BiLSTM with different pre-trained language models: Word2Vec, BERT and RoBERTa. 91

Abbreviations and Notations

The key abbreviations and notations used in this thesis are listed as follows.

DC	Difference of convex functions
DCA	DC algorithm
ML	Machine learning
CD	Coordinate Descent
BCD	Block Coordinate Descent
SVM	Support Vector Machines
LLM	Large Language Model
DL	Deep Learning
CNN	Convolutional Neural Network
LSTM	Long Short-Term Memory
EL	Ensemble learning
NLP	Natural language processing
SGD	stochastic gradient descent
$\mathbb{E}(X)$	expectation of the random variable X
$\mathbb{R}^n$	set of real vectors of size n
$\nabla f(x)$	gradient of f at x
$\partial f(x)$	subdifferential of f at x

Résumé

Les techniques d'apprentissage automatique (ML) jouent un rôle de plus en plus central dans le paysage actuel. Le développement de nouvelles techniques de ML est essentiel, étant donné leur rôle crucial dans divers domaines. Cela englobe la résolution des défis posés par les données à grande échelle et de grande dimension, ainsi que par les données déséquilibrées et la rareté des données. Cette thèse se concentre sur le développement de nouvelles techniques de ML pour résoudre des problèmes pressants dans trois domaines principaux : la gestion des données à grande échelle, la gestion des données déséquilibrées dans le domaine financier et l'atténuation de la rareté des données grâce à l'apprentissage par transfert dans le domaine de la santé. Nos méthodologies de ML proposées reposent sur la programmation DC (Différence de fonctions convexes) et les algorithmes DCA (DC), intégrant et améliorant trois techniques de ML fondamentales : SVM, bagging et l'apprentissage en profondeur par transfert.

La thèse comprend cinq chapitres : Le chapitre 1 sert d'introduction, présentant les concepts fondamentaux de ML, la programmation DC et DCA. Dans le chapitre 2, nous examinons la technique de coordonnées par bloc pour traiter les problèmes à grande échelle dans SVM dans le contexte des mégadonnées. En combinant cette technique avec DCA, nous proposons l'algorithme appelé SVM DCA à coordonnées par bloc (BC-DCASVM), qui permet à l'algorithme SVM à noyau de résoudre les problèmes liés à la grande dimensionnalité. Dans le processus d'apprentissage, l'algorithme met à jour un bloc de coordonnées (dimensions) à la fois pour réduire efficacement la valeur de l'objectif tout en maintenant les autres blocs fixes. Le chapitre 3 se concentre sur l'étude de la technique du bagging et de divers problèmes financiers, suivie de la proposition d'une solution pour relever ces défis. Le premier algorithme, appelé Bagging DCA pondéré (BaggingDCA), vise à résoudre les problèmes qui peuvent survenir lors de l'application du bagging à des problèmes d'apprentissage automatique généraux. Le deuxième algorithme intègre BaggingDCA et des techniques de sensibilité au coût dans l'algorithme de bagging, appelé Bagging DCA pondéré sensible au coût (CSB-DCA). Cet algorithme s'attaque directement à l'un des problèmes les plus difficiles en ML, à savoir les données déséquilibrées, qui affectent également de nombreuses tâches de classification financière. En incorporant BaggingDCA et la technique de sensibilité au coût, l'algorithme vise à réduire le biais induit par le déséquilibre et à améliorer les performances prédictives sur des ensembles de données financières biaisés. Nous concevons les algorithmes de bagging pondéré basés sur DCA pour être polyvalents, permettant l'utilisation de différents apprenants de base et de différentes fonctions de perte dans une conception unifiée. Le quatrième chapitre explore divers problèmes de santé, où les données textuelles médicales sont souvent rares en raison de leur sensibilité. En tirant parti des perspectives prometteuses de l'algorithme de DCA stochas-

tique à chaînes de Markov (MCSDCA), un optimiseur basé sur DCA pour l'apprentissage en profondeur, qui a été évalué sur des architectures d'apprentissage en profondeur traditionnelles, nous proposons une nouvelle architecture d'apprentissage en profondeur qui combine CNN et BiLSTM avec l'algorithme MCSDCA pour relever certains défis cruciaux dans le domaine de la santé. De plus, nous utilisons plusieurs modèles de langage pré-entraînés pour relever le défi de la rareté des données, en nous inspirant des principes de l'apprentissage par transfert. En comparant avec des optimiseurs populaires, des architectures d'apprentissage en profondeur et des modèles de langage pré-entraînés, notre méthode démontre sa compétitivité par rapport aux approches existantes. Enfin, le chapitre 5 sert de conclusion de la thèse, fournissant un résumé complet des principales conclusions et des recommandations pour les orientations futures de la recherche.

Mots-clés: Programmation DC (Différence de fonctions convexes) et algorithmes DCA (DC), SVM à grande échelle, Bagging pondéré, Apprentissage par transfert

Abstract

Machine learning (ML) techniques are assuming an ever more pivotal role in today's landscape. The development of new ML techniques is essential, given their crucial role in diverse domains. This encompasses addressing challenges posed by large-scale and high-dimensional data, as well as imbalanced data and scarcity of data. This thesis focuses on the development of new ML techniques to address pressing issues in three main topics: Addressing large-scale data, handling imbalanced data in the financial domain, and mitigating data scarcity using transfer learning in healthcare. Our proposed ML methodologies are based on DC (Difference of Convex functions) programming and DCA (DC Algorithms), integrating and enhancing three fundamental ML techniques: SVM, bagging, and deep transfer learning.

The thesis comprises five chapters: Chapter 1 serves as an introduction, presenting fundamental concepts of ML, DC programming, and DCA. In Chapter 2, we investigate the block-coordinate technique to handle large-scale problems in SVM within the context of big data. By combining this technique with DCA, we propose the algorithm named Block-Coordinate DCA SVM (BC-DCASVM), which enables the kernel-SVM algorithm to address problems arising from high dimensionality. In the training process, the algorithm updates one block of coordinates (dimensions) at a time to effectively decrease the objective value while keeping the other blocks fixed. Chapter 3 focuses on studying the bagging technique and various financial problems, followed by the proposal of a solution to address these challenges. The first algorithm, called DCA weighted Bagging (BaggingDCA), aims to address issues that can arise when applying bagging to general machine learning problems. The second algorithm integrates BaggingDCA and cost-sensitive techniques into the bagging algorithm, which is named Cost-Sensitive weighted Bagging DCA (CSB-DCA). This algorithm directly tackles one of the most difficult issues in ML—imbalanced data—which also plagues many financial classification tasks. By incorporating BaggingDCA and the cost-sensitive technique, the algorithm aims to reduce imbalance-driven bias and improve predictive performance on skewed financial datasets. We design the DCA-based weighted bagging algorithms to be versatile, allowing the use of various base learners and different loss functions within a unified design. The fourth chapter explores various issues in health care, where medical text data is often scarce due to its sensitivity. Leveraging the promising prospects of the Markov-chain stochastic DCA (MCS-DCA) algorithm, an optimizer based on DCA for deep learning, which has been evaluated on traditional deep learning architectures, we propose a new deep learning architecture that combines CNN and BiLSTM with the MCSDCA algorithm to address some crucial challenges in the healthcare field. Moreover, we employ several pre-trained language models to address the data scarcity challenge, drawing inspiration from the principles of transfer learning. Through comparisons with popular optimizers, deep learning architectures, and pre-trained language models, our method demonstrates competitiveness with existing approaches. Lastly, Chapter 5

serves as the concluding chapter of the thesis, providing a comprehensive summary of the key findings and recommendations for future research directions.

Keywords: DC (Difference of Convex functions) programming and DCA (DC Algorithms), Large-scale SVM, Weighted bagging, Transfer learning

Introduction générale

Cadre général et nos motivations

Au cours des dernières décennies, l'apprentissage automatique (ML) a connu une augmentation sans précédent de son application dans divers domaines, remodelant profondément le paysage de nombreux secteurs de notre vie. Cette expansion est attribuée aux progrès remarquables de la puissance de calcul, à la disponibilité de vastes ensembles de données et à l'amélioration d'algorithmes sophistiqués. La combinaison de ces facteurs a fait de l'apprentissage automatique un outil essentiel pour extraire des informations, faire des prédictions et automatiser des tâches complexes.

Cette tendance actuelle est particulièrement remarquable dans les domaines de la finance et de la santé, où la combinaison des progrès technologiques et de l'expansion rapide des données à grande échelle a conduit à l'apparition de nouvelles opportunités et défis. Ces systèmes d'apprentissage automatique sont désormais déployés avec succès dans de nombreuses applications, notamment la détection de la fraude à la carte de crédit, l'évaluation du crédit client, la prédiction des résultats financiers, ainsi que dans des contextes médicaux, en aidant à l'identification de maladies rares et en soutenant la prise de décision clinique, etc.

La quantité croissante de données dans les domaines de la finance et de la santé, caractérisée non seulement par son ampleur, mais aussi par son bruit inhérent, ses données déséquilibrées ou rareté des données, a conduit à un besoin croissant de nouvelles approches d'apprentissage automatique (ML) innovantes. Ces approches doivent être conçues pour relever le défi des données à grande échelle et résoudre avec succès les problèmes découlant du big data, qui peuvent avoir un impact négatif sur les modèles ML traditionnels.

Cette thèse tente d'explorer et de proposer de nouvelles techniques dans le domaine de l'apprentissage automatique, en abordant les défis uniques posés par les domaines de la finance et de la santé, en particulier à l'ère du big data. Dans ce contexte, la thèse vise à proposer de nouvelles méthodes d'apprentissage automatique en améliorant certaines méthodes conventionnelles. En particulier, nous explorons les complications découlant des données de grande dimension, des ensembles de données contaminés par du bruit, des distributions de classes inégales et de la disponibilité limitée de données étiquetées. En comprenant et en surmontant ces obstacles, nous pouvons poser les bases de solutions d'apprentissage automatique

plus résilientes et efficaces, capables d'exceller dans des situations du monde réel, notamment la méthode de descente de coordonnée par blocs, la méthode de bagging et la méthode d'apprentissage par transfert.

Au cœur de l'efficacité du ML moderne se trouve la combinaison de plusieurs composants, comprenant les données et les modèles d'optimisation. De nombreux algorithmes d'optimisation ont été proposés pour le ML, et DCA est l'une des techniques d'optimisation exceptionnelles qui s'est avérée très efficace dans diverses applications, en particulier l'optimisation non convexe [39, 40, 41, 42]. Cependant, la recherche sur les applications de DCA pour le ML afin de relever les défis de données susmentionnés est encore limitée. Afin de contribuer à cette direction de recherche, la programmation DC et DCA sont les principales méthodologies d'optimisation utilisées dans cette thèse.

Une fonction est dite DC (Difference of Convex) si elle peut être représentée comme la différence entre deux fonctions convexes. Cette propriété conduit à la formulation d'un programme DC comme suit :

$$\alpha = \inf \{f(x) = g(x) - h(x) : x \in \mathbb{R}^n\} \quad (P_{dc})$$

òu g et h sont des fonctions convexes définies sur $\mathbb{R}^n$ et à valeurs dans $\mathbb{R} \cup \{+\infty\}$, semi-continues inférieurement et propres. La fonction f est appelée fonction DC avec les composantes DC g et h, et $g - h$ est une décomposition DC de f. DCA est basé sur la dualité DC et des conditions d'optimalité locale. La construction de DCA implique les composantes DC g et h et non la fonction DC f elle-même. Chaque fonction DC admet une infinité des décompositions DC qui influencent considérablement sur la qualité (la rapidité, l'efficacité, la globalité de la solution obtenue ...) de DCA. Ainsi, au point de vue algorithmique, la recherche d'une "bonne" décomposition DC et d'un "bon" point initial est très importante dans le développement de DCA pour la résolution d'un programme DC. L'utilisation de la programmation DC et DCA dans cette thèse est justifiée par de multiple arguments [76]:

- La programmation DC et DCA offrent un cadre extrêmement diversifié pour les problèmes de ML: ML constituent une mine des programmes DC dont la résolution appropriée devrait recourir à la programmation DC et DCA. En effet, la liste non exhaustive des références dans [43] illustre la vitalité, la puissance et la pertinence de cette approche au sein de la communauté de la ML."

- DCA représente davantage une philosophie qu'un algorithme. Pour chaque problème, il est possible de concevoir une gamme d'algorithmes basés sur les principes du DCA. La souplesse du DCA dans le choix des décompositions DC peut conduire à des schémas DCA qui surpassent les méthodes standards en termes de performance.

- L'analyse convexe offre des instruments puissants pour démontrer la convergence de DCA dans un cadre général. Par conséquent, tous les algorithmes fondés sur DCA prof-

itent (au moins) des propriétés de convergence générales du schéma DCA générique qui ont été démontrées.

- DCA est une méthode efficace, rapide et scalable pour la programmation non convexe. A notre connaissance, DCA est l'un des rares algorithmes de la programmation non convexe, non différentiable qui peut résoudre des programmes DC de très grande dimension.

Il est essentiel de remarquer que grâce à la les techniques de reformulation en programmation DC et à des décompositions DC appropriées, la plupart des algorithmes existants en programmation convexe/non convexe comme cas particuliers de DCA.

Nos contributions

La thèse se concentre principalement sur la conception et l'étude d'algorithmes innovants de ML basés sur la programmation DC et le DCA pour relever de nouveaux défis dans les problèmes de ML du monde réel, en particulier les problèmes à grande échelle, les problèmes financiers déséquilibrés et les problèmes de santé.

D'une part, nous proposons trois algorithmes de ML différents, du ML classique à l'apprentissage en profondeur et à l'apprentissage par transfert, qui peuvent résoudre différents types de problèmes de ML ainsi que différents domaines de problèmes du monde réel. D'autre part, en exploitant le DCA, qui est une puissante technique d'optimisation qui s'est avérée plus performante que d'autres méthodes d'optimisation convexe dans diverses applications, notre algorithme proposé peut ouvrir des approches potentielles à certains problèmes de ML, tels que l'amélioration des performances et la réduction du bruit. Plus précisément, nous contribuons à trois algorithmes différents de la manière suivante :

Tout d'abord, nous proposons une nouvelle méthode basée sur le DCA avec coordination par blocs pour résoudre des programmes DC de grande dimension. Ensuite, la méthode proposée est spécifiquement appliquée à la SVM à noyau à grande échelle de manière unifiée, capable de traiter presque toutes les pertes couramment utilisées dans la SVM, y compris les pertes convexes et non convexes, ainsi que celles pour la classification et la régression. Grâce à l'approche de coordination par blocs avec une décomposition DC appropriée, l'algorithme proposé peut gérer efficacement des ensembles de données avec une grande dimensionnalité et un grand nombre d'instances tout en maintenant des performances compétitives, ce qui est souvent un compromis pour les algorithmes existants à grande échelle. Ainsi, notre méthode peut surmonter les goulots d'étranglement de stockage et de calcul des SVM classiques lorsqu'il s'agit de traiter de grandes quantités de données. Cependant, pour traiter des problèmes à grande échelle, tandis que d'autres méthodes utilisent des techniques d'approximation, telles que la méthode de faible rang, qui peut être une limitation lors du traitement d'ensembles de données non fortement corrélés, notre solution utilise l'approche de coordination par blocs qui

est indépendante de la corrélation des données. Du point de vue théorique, l'algorithme proposé garantit de trouver un point critique DC faible. Cette approche a le potentiel d'améliorer les performances des SVM dans diverses applications telles que la classification d'images, le traitement du langage naturel et la bioinformatique dans le contexte des mégadonnées.

La deuxième contribution de la thèse est la proposition de nouveaux algorithmes d'emballage (bagging) basés sur le schéma LS-DC, qui est applicable à la majorité des fonctions de perte existantes pour les problèmes de classification et de régression. Le premier algorithme proposé, BaggingDCA, hérite des avantages de l'emballage standard, tels que la réduction de la variance, la possibilité de mettre en place des calculs parallèles et l'aide à la prévention du surajustement, en améliorant en outre la précision du modèle d'entraînement. Nous étendons BaggingDCA pour traiter spécifiquement le déséquilibre de classe prévalent dans les problèmes de classification financière. De plus, il est connu que les pertes non convexes, telles que la perte en rampe, sont plus résistantes aux valeurs aberrantes que les pertes convexes. En utilisant la perte en rampe avec le réglage du DCA, nos approches proposées améliorent encore les performances de l'emballage, ce qui augmente la diversité des modèles de l'ensemble et améliore leur robustesse aux données bruitées.

Troisièmement, notre thèse apporte une contribution en introduisant un algorithme d'apprentissage en profondeur hybride CNN-LSTM avec des modèles de langage pré-entraînés tels que Word2Vec, BERT, RoBERTa en conjonction avec le DCA stochastique de chaîne de Markov (MCSDCA).Alors que le MCSDCA s'est avéré être une méthode permettant d'améliorer les performances de certains modèles classiques d'apprentissage en profondeur tels que CNN et LSTM pour certains problèmes de séries chronologiques, le transfert d'apprentissage est une technique d'apprentissage automatique qui utilise les connaissances des modèles pré-entraînés pour les tâches ultérieures afin d'améliorer les performances et le temps d'entraînement de ces modèles. De plus, la combinaison du CNN et du LSTM peut aider le modèle de fusion à apprendre des relations plus complexes entre les mots et à comprendre plus précisément le sens du texte. Notre étude propose également une évaluation complète de l'algorithme MCSDCA sur différentes architectures avancées d'apprentissage en profondeur. Cette approche offre un moyen plus efficace de développer des modèles de traitement du langage naturel (NLP) pour les applications de santé, qui peuvent être utilisés pour diverses tâches telles que le diagnostic, la planification du traitement et la surveillance des patients.

Organisation de la thèse

La thèse se structure en cinq chapitres. Le Chapitre 1 présente des connaissances fondamentales sur l'apprentissage automatique (ML), les méthodologies ML utilisées et introduit la programmation DC et le DCA. Dans le Chapitre 2, nous examinons la méthode de coordination par blocs, une stratégie pour résoudre les défis du ML en grande dimension. Ensuite, une approche basée sur le DCA avec coordination par blocs est proposée pour la SVM à noyau. Le

Chapitre 3 explore l'apprentissage ensembliste, ce qui conduit à la création de deux algorithmes : BaggingDC et CSB-DCA adaptés aux problèmes financiers déséquilibrés. Le Chapitre 4 se penche sur la méthode MCSDCA, appliquée à l'apprentissage par transfert dans le traitement du langage naturel et les problèmes de santé. Enfin, le Chapitre 5 conclut la thèse.

Chapter 1

Methodologies and Fundamentals

1.1 Machine learning and challenges

In recent years, the field of machine learning (ML) has experienced remarkable growth and has become an essential component in various aspects of daily life. Particularly impactful domains such as finance and healthcare have witnessed the successful integration of ML methodologies. These ML systems are now deployed across numerous applications, including the detection of credit fraud, customer credit approval, financial outcome prediction [18, 20, 102], and within medical contexts, aiding in the identification of rare diseases and supporting clinical decision-making [1, 12].

The recent achievements of these ML algorithms are rooted in their capacity to identify patterns within large, complex datasets and effectively extend these patterns to new, unseen data instances. The growth of data caused by social networks, cloud applications, IoTs, etc. has created new challenges and opportunities in data processing and analysis. On the one hand, large-scale data gives ML algorithms a vast quantity of data to analyze and process, thereby enhancing the quality of predictions. On the other hand, large-scale data also makes it challenging for traditional ML algorithms to efficiently exploit and process. Furthermore, some challenges, such as data imbalance, noise, and data scarcity, are also difficulties that algorithms face in many important fields. The healthcare and finance sectors are no exception. Data in these sectors tends to be highly sensitive, and errors could have severe consequences. Specifically in healthcare and finance, state-of-the-art algorithms are still needed to safely and accurately analyze private data and make predictions while avoiding potential harms that could come from failures or limitations in current ML techniques. Promoting further research aimed at solving challenges specific to these consequential domains is therefore still very urgently needed. This thesis aims to take meaningful steps toward addressing these important challenges. Through developing innovative ML algorithms, this endeavor seeks to contribute to resolving these urgent problems.

The purpose of this chapter is to provide an introductory overview of ML and its essential

methodologies within the context of big data. Here, we provide an introductory glimpse into the world of ML, highlighting its challenges, fundamental concepts, and methodological techniques. These elements collectively lay the foundation for the new algorithms we propose and develop in this thesis.

1.1.1 Machine learning

ML is programming computers to optimize a performance criterion using example data or past experience [2]. According to this interpretation, data and optimization algorithms are the two crucial components of ML. The optimization algorithm, which serves as the core element of most ML algorithms, works by processing the input data and iteratively adjusting its performance on a given task in order to improve its effectiveness over sequential iterations. Through this optimization procedure, the ML model enhances its ability to make data-related predictions, classifications, and other decisions.

Let $\mathcal{D}$ be the dataset consisting of input-output pairs $(\mathbf{x}_i, y_i)$ where $\mathbf{x}_i \in \mathbb{R}^n$ represents the input features and $y_i \in \mathbb{R}$ is the corresponding output or label. The goal of ML is to learn a function $f : \mathbb{R}^n \to \mathbb{R}$ that accurately maps input features to outputs. The approach involves searching for a model h within a hypothesis space $\mathcal{H}$ that minimizes the expected loss over the data.

$$h^* = \arg\min_{h \in \mathcal{H}} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[L(h(\mathbf{x}), y)]$$

In this context, the function $L(h(\mathbf{x}), y)$ quantifies the discrepancy between the predicted output $h(\mathbf{x})$ and the true label y. However, in practice, we optimize the empirical risk minimization instead:

$$h^* = \arg\min_{h \in \mathcal{H}} \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} L(h(\mathbf{x}_i), y_i)$$

The crux of the matter lies in estimating the function f using a hypothesis h from a predefined family of functions $\mathcal{H}$. This is achieved by minimizing a suitable loss function $L(h(\mathbf{x}_i), y_i)$ over the dataset $\mathcal{D}$. The fundamental idea emphasizes that the effectiveness of a resulting ML model is closely linked to the characteristics of the readily available training data, encompassing the quantity, quality, and variety of the data, as well as an adequate optimization algorithm.

1.1.2 Data's challenges

It is obvious that the efficacy of ML algorithms is highly dependent on the quantity, quality, and relevance of the data. Insufficient or biased data may lead to poor generalization and overfitting. In the context of big data, where datasets are extensive and diverse, there are numerous

challenges of data, such as large-scale data, noisy data, imbalanced data, and data scarcity problems. These issues necessitate the continuous development of new advanced algorithms to fully leverage large and complex real-world data.

Large-scale of data

The term "large-scale data" refers to the vast amounts of information generated from various sources such as enterprise databases, social media, sensors, online transactions, and more. This data flood presents both challenges and opportunities for businesses, researchers, and organizations across different sectors. Having more data for a machine learning model is undoubtedly advantageous. Consider, for example, a machine learning algorithm that predicts stock prices using historical data. If the algorithm has access to daily stock price data for hundreds of companies spanning multiple years, it will be better able to identify market trends and relationships. The number of instances, or data points, is significant because it enables the algorithm to understand patterns and relationships more effectively. With a larger, higher-quality dataset, the algorithm has a greater opportunity to generalize and make more accurate predictions.

However, having abundant data also introduces an abundance of challenges for ML algorithms. In these situations, the "curse of dimensionality" becomes increasingly relevant. The concept of "large-scale data" isn't just about the quantity of samples; it's also influenced by the complexity of the features in the data. As the number of features grows, the data becomes sparse in the high-dimensional space, making it difficult for algorithms to find meaningful patterns or relationships. This sparsity can result in overfitting [70], where models memorize noise in the data rather than capturing true underlying trends. Moreover, the curse of dimensionality can result in increased computational complexity and resource requirements, which substantially slows down training and inference times. Despite the abundance of available data, the enormous number of features can degrade the quality of extracted insights and increase the chance of dealing with noisy, irrelevant, or redundant attributes.

Numerous proposed studies address the challenges of managing large-scale datasets. For instance, techniques like Principal Component Analysis (PCA) [99] can reduce dimensionality by selecting crucial features, optimizing model performance. Strategies such as Kernel Trick, Approximation, Parallelization, and Distributed Computing enhance efficiency and tackle complexities in high-dimensional data [8, 81, 111], Feature Selection and Sparse Models manage dimensionality and computational loads [40, 45]. However, there is no one-size-fits-all solution for the intricacies of large-scale and high-dimensional data. The choice of approach depends on data specifics and problem objectives. In the future, continuous research and innovative methods will remain pivotal to harnessing the potential of vast datasets in ML and data science, aiming to optimize their utilization effectively.

Noise data

In the context of big data, accompanied by the surge in massive-scale data, the issue of data noise has come to the forefront. Data noise refers to unwanted, inaccurate, or incomprehensible values that can appear within a dataset [101]. This can result from factors such as errors in data collection, environmental interference, or inherent variations in the data. Data noise, can lead to negative impacts on data analysis and ML tasks by distorting analytical outcomes with misleading and erroneous information. If not carefully addressed, data noise can cause ML models to overfit, where the model overly focuses on learning from noise rather than understanding true patterns and relationships.

To handle data noise, in addition to working directly with the noisy data through preprocessing steps and data cleansing, such as removing noisy values or imputing missing values, one can employ ML models with "noise resistance" capabilities. For example, Support Vector Machines (SVM) with loss functions designed to counter noise can help improve the model's generalization ability towards noisy data [108]. Moreover, techniques like regularization can be employed to control the complexity of the model and prevent it from fitting noise too closely [90]. This ensures that the model captures meaningful patterns while disregarding noise.

Imbalanced data

Imbalanced data presents a significant challenge in ML, that requires cautious consideration due to its far-reaching implications. Imbalanced data refers to datasets in which the distribution of classes is significantly uneven, a common phenomenon observed in numerous real-world situations. Several factors contribute to the emergence of imbalanced data. Real-world circumstances, such as infrequent occurrences or minority groups (Fraud detection, credit approval problem in the finance domain), frequently result in skewed class distributions. When data acquisition is subject to biases or limitations, the collection process itself can also accidentally result in imbalanced datasets.

The consequences of working with imbalanced data are far-reaching. As a result of their bias towards the majority class, ML models trained on such datasets frequently struggle to accurately predict the minority class. This can lead to suboptimal performance and skewed decision boundaries, resulting in misclassification of the minority class instances.

In order to resolve the complexities posed by unbalanced data, researchers have employed a variety of techniques. These techniques include methods of direct data manipulation as well as enhancements to ML models in order to mitigate the negative effects of imbalanced data. Solutions to the issue of imbalanced data include oversampling the minority class, undersampling the majority class, and employing sophisticated algorithms such as SMOTE (Synthetic Minority Oversampling Technique) [19]. Furthermore, ensemble methods like Random Forest [15] and AdaBoost [27] have shown efficacy in handling imbalanced data. These techniques construct a combination of multiple models, often providing improved generalization and miti-

gating the bias caused by skewed class ratios. Additionally, cost-sensitive learning [25] assigns varying misclassification costs to different classes, encouraging the model to focus on accurate predictions for the minority class, thereby rectifying imbalanced data's impact.

Data scarcity

The paradox of data scarcity presents itself as a confusing obstacle in the context of big data, where our digital landscape has become overloaded with an abundance of information. This puzzle, which contradicts the prevailing narrative of data abundance, requires a more profound investigation. Delving into the heart of this paradox, several complex factors come to the fore, shedding light on the underlying causes for the lack of suitable training data.

First and foremost, the overwhelming volume of available data can be misleading. Among all the vast information out there, it's actually quite hard to find data that's both relevant and well-organized for specific ML tasks. Take supervised machine learning models, for example. They need data that's labeled with precise annotations and categories. However, this labeling process requires humans to intervene, and it takes a lot of time. This slows down the progress of artificial intelligence. Lots of data exists, but it often does not match what we need for specific learning goals, causing this scarcity problem. Secondly, concerns related to data privacy and ethics impose limitations on data accessibility. With growing emphasis on user privacy and data rights, obtaining comprehensive and unrestricted datasets becomes intricate. This ethical aspect introduces obstacles to the collection of extensive real-world data, preventing the smooth acquisition required for efficient machine learning model development. Lastly, the evolving nature of tasks demands the obtaining of specialized data, which is frequently unavailable due to novelty. Emerging domains such as medical diagnostics or predictions of rare phenomena suffer from insufficient historical data, impeding the training process. For instance, in the field of healthcare, obtaining abundant labeled datasets for training on certain types of rare cancers like eye cancer [78] or endocrine cancer proves to be challenging.

Consequently, the paradox of data scarcity highlights the necessity for innovative approaches, such as transfer learning to address this challenge and lead ML into a new era of progress.

1.2 Prerequisite knowledge for our methods

In this section, we will introduce fundamental ML and optimization techniques. These techniques serve as the foundation for the development of novel algorithms in later chapters, which address the challenges inherent to the aforementioned data scenarios.

1.2.1 Coordinate Descent method

Coordinate Descent (CD) [100] is an optimization technique used to solve large-scale optimization problems that successively minimizes along coordinate directions or coordinate hy-

perplanes to find the minimum of a function. During each iteration, the algorithm selects a coordinate using the coordinate selection rule, then exactly or inexactly minimizes over the corresponding coordinate hyperplane while keeping all other coordinates fixed.

Given a objective function that need to be minimized $f(\mathbf{x})$ where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. The optimization problem can be formulated as:

$$\min_{\mathbf{x}} f(\mathbf{x}) = f(x_1, x_2, \ldots, x_n),$$

subject to relevant constraints. At each iteration k, an index $i_k \in \{1, 2, \ldots, n\}$ is selected, and the decision vector is updated to approximately minimize the objective function along the i_k-th coordinate. The technique exhibits either deterministic or randomized behavior, contingent upon the selection of the update coordinates. If the coordinate indices i_k are chosen sequentially from the set $\{1, 2, \ldots, n\}$, then the approach is denoted as the Cyclic Coordinate Descent (CCD) method. Conversely, if i_k is uniformly sampled from $\{1, 2, \ldots, n\}$, the resulting technique is termed the Randomized Coordinate Descent (RCD) method [29]. The CD method iteratively updates each coordinate x_j while keeping the others fixed. The update rule for the $i - th$ coordinate in the k-th iteration can be written as:

$$x_j^{(k+1)} = \arg\min_{x_j} f(x_1^{(k)}, x_2^{(k)}, \ldots, x_{j-1}^{(k)}, x_j, x_{j+1}^{(k)}, \ldots, x_n^{(k)})$$

The algorithm continues iteratively until convergence is achieved, indicating that further updates to $\mathbf{x}$ do not significantly affect the function value. The CD algorithm is presented in Algorithm 1 as follows:

Algorithm 1 Coordinate Descent algorithm

1. Initialize $\mathbf{x}^0 = (x_1^0, x_2^0, \ldots, x_n^0)$ with initial guesses for each coordinate.
2. For $k = 0, 1, 2, \ldots$:
   (a) Choose a coordinate index $i_k \in \{1, 2, \ldots, n\}$. Fix all coordinates except x_i: $\mathbf{x}_{-i}^k = (x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ at iteration k.
   (b) Update x_i by minimizing f over the x_i-axis while keeping $\mathbf{x}_{-i}^k$ constant:
   $$x_i^{k+1} = \arg\min_{x_i} f(\mathbf{x}_i, \mathbf{x}_{-i}^k).$$
   (c) Repeat step (b) for all coordinates.
3. Continue the iterations until a stopping criterion is met.

The optimization problem in step 2(b) of the Algorithm 1 can typically be solved for many types of functions, making the CD method efficient.

Let's illustrate the CD algorithm applied to the quadratic function $f(x, y) = 3x^2 - 4xy + 5y^2$.

We aim to minimize this function using the CD method described earlier. The process of the function is illustrated in Figure 1.1 as follows:

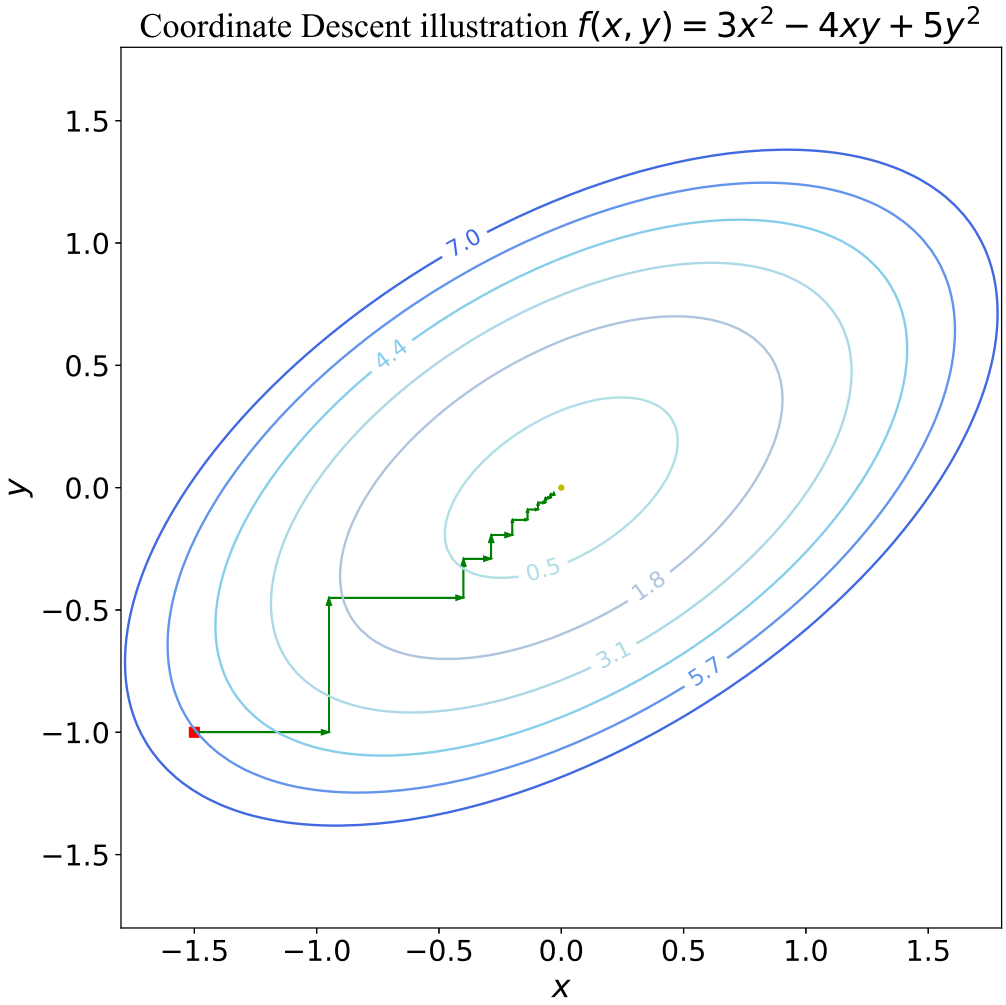


Figure 1.1: Illustrate the Coordinate Descent (CD) method applied to the function $f(x, y) = 3x^2 - 4xy + 5y^2$, demonstrating the step-by-step iterative updates along the x and y coordinates, leading to the approach of the minimum value. Starting from the point $(x, y) = (-1.5, -1)$ and first performing line-search along the x-axis, the CD algorithm alternates between updating the x coordinate and the y coordinate in each iteration. This iterative process continues until it reaches the minimum value at $f(0, 0) = 0$ (or stop criterion is met).

Block Coordinate Descent method

The Block Coordinate Descent (BCD) method is an extension of the CD method. By dividing the variables into non-overlapping blocks, the BCD method introduces more flexibility in terms of updating variables jointly, which can lead to improved convergence in some cases. Recent studies [65, 7] have demonstrated the efficacy of BCD as a suitable method for addressing

challenges posed by large-scale datasets.

In each iteration, this method selects a block of variables and updates them jointly while keeping the other blocks fixed. The choice of block selection strategy, whether sequential or randomized, affects the behavior of the algorithm. The update rule can be written as:

$$x_{\text{block}}^{(k+1)} = \arg\min_{x_{\text{block}}} f(x_{\text{fixed}}^{(k)}, x_{\text{block}})$$

where $x_{\text{fixed}}^{(k)}$ represents the variables in the fixed blocks. This process is repeated for each block until convergence.

The BCD method can be outlined in the Algorithm 2:

Algorithm 2 Block Coordinate Descent algorithm

1. Initialize $\mathbf{x}^0 = (x_1^0, x_2^0, \ldots, x_n^0)$ with initial guesses for each coordinate.
2. Specify the number of blocks and divide the variables into non-overlapping blocks.
3. Initialize iteration counter $k = 0$.
4. **While** convergence criteria not met **do**:
   (a) Select a block x_{block} for update (e.g. Randomized or Sequential).
   (b) Fix the variables in the other blocks as $x_{\text{fixed}}^{(k)}$.
   (c) Update the selected block by solving:
   $$x_{\text{block}}^{(k+1)} = \arg\min_{x_{\text{block}}} f(x_{\text{fixed}}^{(k)}, x_{\text{block}})$$
   (d) Update $k = k + 1$.
5. **End While**

1.2.2 Ensemble learning and Bagging

Ensemble learning [14, 83] is an ML technique that combines multiple individual models, known as base models or weak learners, to create a more robust and accurate prediction model. It leverages the concept of "wisdom of the crowd" by aggregating the predictions of multiple models (whether these models employ the same or different algorithms) to make a final prediction. This technique has gained popularity due to its ability to tackle complex problems and enhance model accuracy in various domains, including classification, regression, and even anomaly detection [5, 58, 85, 97].

The illustration of ensemble learning is in Figure 1.2 as follows:

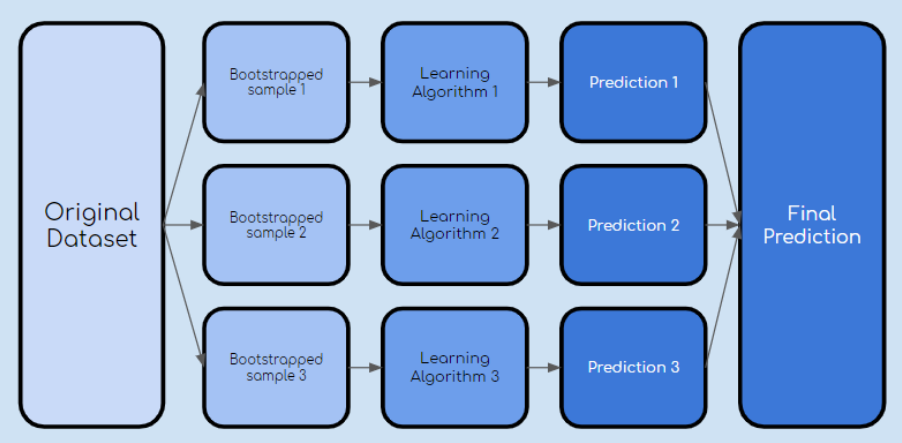


Figure 1.2: Illustration of ensemble learning (bagging). Each base model takes a subset of samples from the original dataset. The base models are then trained sequentially or in parallel to produce diverse predictions. The ensemble model combines the predictions of the base models, usually by taking a majority vote or averaging, to make the final prediction.

Consider a supervised learning problem where we have a training dataset $\mathcal{D}$ consisting of N samples, each with M features. The dataset can be represented as

$$\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\},$$

where $x_i \in \mathbb{R}^M$ represents the feature vector for the ith sample, and y_i is the corresponding target label. In an ensemble, we have N base models indexed by i, each producing a prediction denoted as $f_i(X)$. The ensemble prediction $F(X)$ is a combination of these base model predictions. For aggregating the predictions of base models to make a final prediction, common ensemble methods include:

- **Voting**: Each base model's prediction contributes equally.

$$F(X) = \text{mode}(f_1(X), f_2(X), \ldots, f_N(X))$$

- **Averaging**: Averaging the predictions of base models.

$$F(X) = \frac{1}{N} \sum_{i=1}^{N} f_i(X)$$

- **Weighted Averaging**: Averaging with different weights w_i.

$$F(X) = \sum_{i=1}^{N} w_i \cdot f_i(X)$$

Ensemble learning can be broadly categorized into three main types: Bagging, Boosting, and Stacking, each designed is to enhance predictive performance through model combination.

Bagging builds diverse base models by training them independently on bootstrapped subsets of data and aggregates their predictions, reducing variance and enhancing stability. In Boosting, weak models are iteratively refined by assigning weights to correct previous mistakes, ultimately creating a strong learner with improved accuracy. Stacking, on the other hand, employs a meta-model to integrate base models' predictions, capturing a wide range of predictive behaviors. In this thesis, the focus will be on the Bagging method.

Bagging

One of the most popular ensemble learning methods is Bagging, which stands for Bootstrap Aggregating. Bagging involves creating multiple subsets of the original training data through random sampling with replacement. Each subset is used to train a separate base model, and the predictions from these models are combined to produce the final ensemble prediction by using an aggregating method (e.g. voting, weighted average). The main idea behind Bagging is to introduce diversity among the base models by training them on different subsets of the data. This helps to reduce the variance of the overall ensemble model and improves its generalization performance. By combining the predictions of multiple models, Bagging aims to achieve a more accurate and stable prediction compared to individual models. In Bagging, multiple base models are trained independently on different bootstrapped subsets of the training data. The final prediction is then obtained by averaging (for regression) or aggregating (e.g. majority voting) (for classification) the predictions of these base models:

$$F_{\text{bagging}}(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x) \quad \text{(for regression)}$$

$$F_{\text{bagging}}(x) = \text{Agregate}\{f_1(x), f_2(x), \ldots, f_N(x)\} \quad \text{(for classification)}$$

The steps involved in the Bagging ensemble learning process are as follows:

1. Data sampling: Random subsets of the original training data are created through bootstrap sampling. Bootstrap sampling involves randomly selecting samples from the training data with replacement, which means that some samples may appear multiple times in a subset, while others may not be included.

2. Base model training: Each subset of the data is used to train a separate base model, typically using the same learning algorithm or a set of similar algorithms. The base models can be decision trees, neural networks, support vector machines, or any other suitable learning algorithm.

3. Prediction aggregation: Once the base models are trained, they are used to make predictions on the test data or new unseen data. The predictions from each base model are combined using a specific aggregation method, such as majority voting for classification problems or averaging for regression problems. The aggregated predictions form the final ensemble prediction.

The standard setting of the Bagging method is presented in the algorithm 3 as follows:

Algorithm 3 Bagging algorithm

Input: Training dataset D, number of base models N
for $i = 1$ **to** N **do**
1. Sample a bootstrap dataset D_i from D
2. Train base model f_i on D_i

end for
3. Combine predictions of base models to get $F(X)$
Output: Ensemble model F

1.2.3 Deep learning and Transfer learning

Transfer learning is a research problem in ML that allows knowledge gained from one task or domain to be transferred and applied to another related task or domain. It leverages the idea that the learned knowledge or representations in one domain can be useful in solving a different but related problem. Instead of training a model from scratch for a new task, transfer learning enables the reuse of pre-trained models or features, saving time and computational resources.

Motivating the study of transfer learning is the ability of humans to intelligently apply previously acquired knowledge to the solution of new problems in a more effective manner or with superior solutions. Consider the case of two individuals who desire to study the piano. One person has no previous experience playing music, whereas the other has a broad knowledge of music through playing the guitar. By transferring previously acquired musical knowledge to the task of learning the piano, a person with an extensive musical background can learn the piano more efficiently. Intuitive about transfer learning is presented in the figure 1.3:

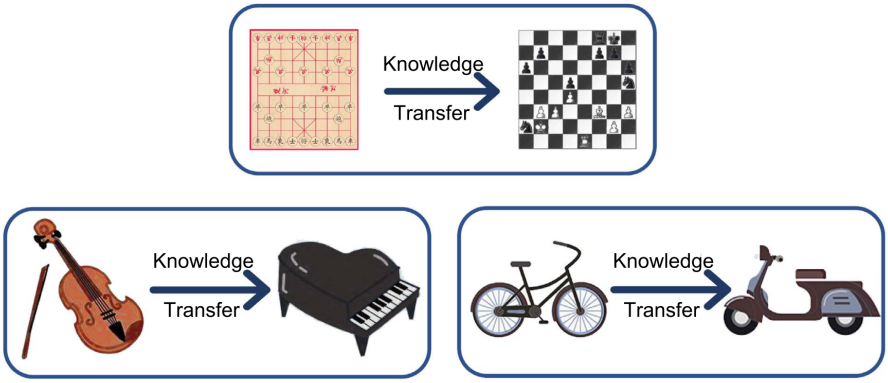


Figure 1.3: Intuitive about transfer learning [113]. Taking inspiration from how people share knowledge between different areas, transfer learning seeks to use what we know from one related field to make learning better in a different area.

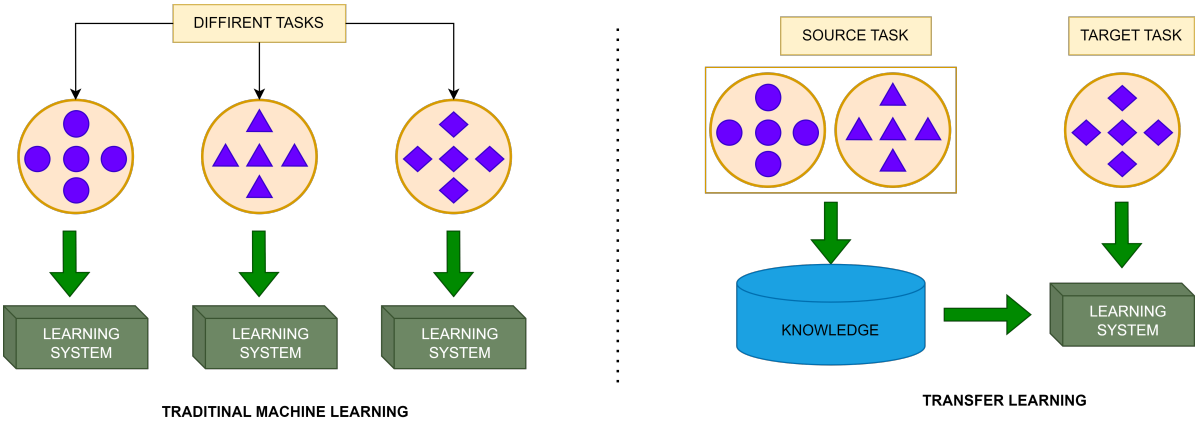


Figure 1.4: Comparision of the learning process between Traditional ML and Transfer learning. While Traditional ML constructs models from scratch using the available dataset and domain-specific features, Transfer learning aims to extract the knowledge from one or more source tasks and applies the knowledge to a target task.

Transfer learning definition

Technically, transfer learning [98] is defined as follows: A domain $\mathcal{D}$ is characterized by two components: a feature space $\mathcal{X}$ and a marginal probability distribution $\mathrm{P(X)}$. Here, $\mathrm{X} = \{\mathrm{x_1}, \ldots, \mathrm{x_n}\} \in \mathcal{X}$. For a given domain $\mathcal{D}$, a task $\mathcal{T}$ comprises a label space $\mathcal{Y}$ and a predictive function $\mathrm{f}(\cdot)$ learned from feature-label pairs $\{x_i, y_i\}$, where $x_i \in X$ and $y_i \in \mathcal{Y}$. Thus, $\mathrm{D} = \{\mathcal{X}, \mathrm{P(X)}\}$ and $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$. The source domain data is defined as $\mathrm{D_S} = \{(\mathrm{x_{S1}}, \mathrm{y_{S1}}) \ldots, (\mathrm{x_{Sn}}, \mathrm{y_{Sn}})\}$, where $\mathrm{x_{Si}} \in \mathcal{X}_S$ is the ith data instance of $\mathrm{D_S}$ and $\mathrm{y_{Si}} \in \mathcal{Y}_S$ is the corresponding class label for $\mathrm{x_{Si}}$. Similarly, the target domain data is defined as $\mathrm{D_T} = \{(\mathrm{x_{T1}}, \mathrm{y_{T1}}) \ldots, (\mathrm{x_{Tn}}, \mathrm{y_{Tn}})\}$, where $\mathrm{x_{Si}} \in \mathcal{X}_S$ is the ith data instance of $\mathrm{D_S}$ and $\mathrm{y_{Si}} \in \mathcal{Y}_S$ is the corresponding class label for $\mathrm{x_{Si}}$. We denote the source task as $\mathcal{T}_S$ with predictive function $\mathrm{f_S}(\cdot)$, and the target task as $\mathcal{T}_\mathcal{T}$ with predictive function $\mathrm{f_T}(\cdot)$.

Given a source domain $\mathcal{D}_\mathrm{S}$ with a corresponding source task $\mathcal{T}_S$ and a target domain $\mathcal{D}_\mathrm{T}$ with a corresponding task $\mathcal{T}_\mathcal{T}$, transfer learning enhances the target predictive function $\mathrm{f_T}(\cdot)$ by incorporating knowledge from the related source domain $\mathcal{D}_\mathrm{S}$ and source task $\mathcal{T}_S$. Depending on the distinctions between the source domain and target domain, source task and target task, various forms of transfer learning can be categorized. In comparison with traditional ML, the knowledge transfer process from source task to target task of Transfer learning is presented in figure 1.4.

Categorization

Transfer learning can be beneficial in several scenarios on various applications, such as image classification, object detection, and image segmentation in computer vision or tasks like sentiment analysis, text classification, and machine translation in NLP. Firstly, when the amount of labeled data for the target task is limited [34], transfer learning allows us to utilize knowledge

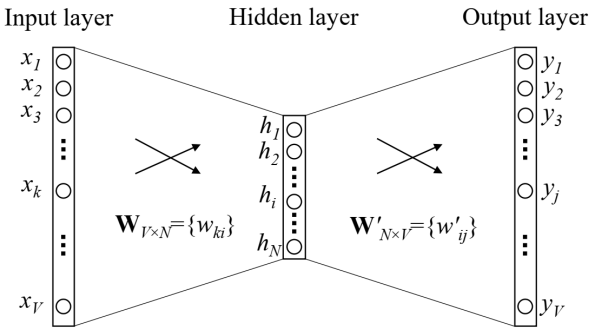


Figure 1.5: Word2Vec architecture (Extracted from [84]).

from a different but related source task with more abundant data (e.g. in Zero-shot Learning, One-shot Learning, Self-Supervised Learning). Secondly, when the target task and source task share similar underlying features or patterns, transfer learning helps in leveraging the pre-existing knowledge to improve performance on the target task [91]. Lastly, even in cases where the target and source tasks are not directly related (aka. Domain adaptation), transfer learning can still provide useful initializations or insights that accelerate the learning process[106].

In recent years, deep neural networks have been widely employed for transfer learning. Consequently, numerous strategies have emerged for implementing transfer learning using deep learning models, including prominent techniques: **Feature-based** transfer learning and **fine-tuning**. In feature-based transfer learning, the pre-trained model is used as a fixed feature extractor, where the early layers are frozen, and only the later layers are modified and trained for the target task. Fine-tuning, on the other hand, involves unfreezing and retraining some or all of the pre-trained model's layers on the target task, allowing the model to adapt to the specific characteristics of the new task. In the domain of Natural Language Processing (NLP), both of these methods have gained widespread adoption across various applications, often employing pre-trained language models.

Pre-trained Language Model

The majority of available datasets for NLP applications are relatively small. This is especially evident in fields such as healthcare, where it is difficult to acquire a large number of annotated datasets for uncommon diseases. This lack of data presents a significant obstacle when training deep neural networks, as they tend to overfit to the training data and have difficulty to generalize to these limited datasets. For many years, pre-trained models with the ImageNet dataset (e.g. VGG-16[89], Resnet[32]) have been standard practice in computer vision. This approach involves imparting the model with foundational knowledge of image features, which can subsequently be applied to diverse visual tasks like image classification and segmentation, yielding outstanding outcomes. Pre-trained language models, which function as the "ImageNet" equivalent in the NLP domain, are a comparable concept in NLP. It is an artificial intelligence model designed to comprehend and generate human language, which has been trained on a

massive amount of text data on a specific architecture of the neural network and is ready to be fine-tuned for particular NLP tasks. For example, while GPT[80] and its variants excel in question-answering tasks, BERT[24] and its variants have gained widespread popularity as a multi-purpose model for a variety of downstream NLP tasks, such as text summarization, data categorization, and question answering. The neural network used for the pre-training process can be as simple as in Word2Vec [60] illustrated in Figure 1.5 or can be sophisticated in terms of architecture like BERT in Figure 1.6.

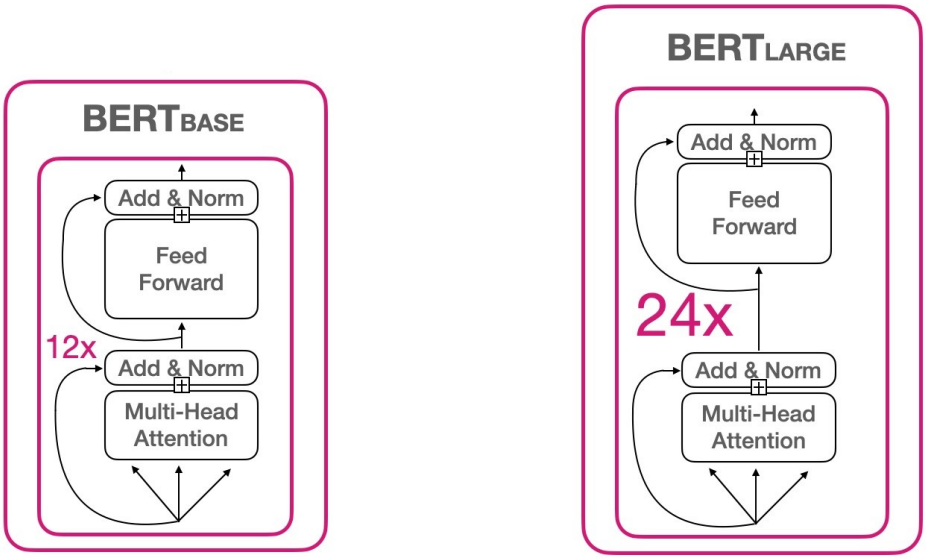


Figure 1.6: Architecture of BERT_Base and BERT_Large (Extracted from HuggingFace[1]). BERT_Base comprises 12 layers of Transformer encoders, a specialized neural network architecture, whereas BERT_Large incorporates 24 layers of Transformer encoders.

1.3 DC Programming and DCA

DC programming, along with the DC Algorithm (DCA), which represent the backbone of nonconvex programming and global optimization, were introduced in a preliminary form by Pham Dinh Tao in 1985 [72]. These concepts have undergone notable enhancements from 1994 onwards, thanks to the collaborative efforts of Le Thi Hoai An and Pham Dinh Tao. This algorithmic approach remains a prominent and widely acknowledged solution for navigating the complexities of nonconvex optimization challenges, as exemplified in references such as [43, 76].

This section introduces the fundamental DC optimization, as well as the standard DC Algorithm upon which the computational methods of this thesis rely. The content presented in this section has been sourced from the publication referenced as [43, 73].

[1]HuggingFace :https://huggingface.co/blog/bert-101

1.3.1 Fundamental convex analysis

This section's objective is to explore key concepts and findings in convex analysis and nonsmooth analysis that hold crucial significance for this thesis.

Denote X the Euclidean space $\mathbb{R}^n$, a subset C of X is said to be *convex* if $(1-\lambda)x + \lambda y \in C$ whenever $x, y \in C$ and $\lambda \in [0,1]$.

Let f be a function whose values are in $\overline{\mathbb{R}}$ and whose domain is a subset S of X. The set

$$\{(x,t) : x \in S, t \in \mathbb{R}, f(x) \leq t\}$$

is called the *epigraph* of f and denoted by epi$f$.

We define f to be a *convex function* on S if epi$f$ is a convex set in $X \times \mathbb{R}$. This is equivalent to that S is convex and

$$f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y), \quad \forall x, y \in S, \forall \lambda \in [0,1]$$

The function f is *strictly convex* if the inequality above holds strictly whenever x and y are distinct in S and $0 < \lambda < 1$.

The *effective domain* of a convex function f on S, denoted by $\operatorname{dom} f$, is the projection on X of the epigraph of f

$$\operatorname{dom} f = \{x : \exists t \in \mathbb{R}, (x,t) \in \text{ epi} f\} = \{x \mid f(x) < +\infty\}$$

and it is convex.

The convex function f is called *proper* if $\operatorname{dom} f \neq \emptyset$ and $f(x) > -\infty$ for all $x \in S$.

The function f is said to be *lower semi-continuous* at a point x of S if

$$f(x) \leq \liminf_{y \to x} f(y)$$

Denote by $\Gamma_0(X)$ the set of all proper lower semi-continuous convex function on X.

Let $\rho \geq 0$ and C be a convex subset of X. One says that a function $\theta : C \mapsto \mathbb{R} \cup \{+\infty\}$ is ρ-convex if

$$\theta[\lambda x + (1-\lambda)y] \leq \lambda\theta(x) + (1-\lambda)\theta(y) - \frac{\lambda(1-\lambda)}{2}\rho\|x-y\|^2$$

for all $x, y \in C$ and $\lambda \in (0,1)$. It is equivalent to say that $\theta - (\rho/2)\|\cdot\|^2$ is convex on C. The modulus of strong convexity of θ on C, denoted by $\rho(\theta, C)$ or $\rho(\theta)$ if $C = X$, is given by

$$\rho(\theta, C) = \sup\left\{\rho \geq 0 : \theta - (\rho/2)\|\cdot\|^2 \text{ is convex on } C\right\}$$

One says that θ is strongly convex on C if $\rho(\theta, c) > 0$. A vector y is said to be a subgradient of

a convex function f at a point x^0 if

$$f(x) \geq f\left(x^0\right) + \left\langle x - x^0, y \right\rangle, \quad \forall x \in X$$

The set of all subgradients of f at x^0 is called the subdifferential of f at x^0 and is denoted by $\partial f\left(x^0\right)$. If $\partial f(x)$ is not empty, f is said to be *subdifferentiable* at x.
For $\varepsilon > 0$, a vector y is said to be a *ε-subgradient* of a convex function f at a point x^0 if

$$f(x) \geq \left(f\left(x^0\right) - \varepsilon\right) + \left\langle x - x^0, y \right\rangle, \quad \forall x \in X$$

The set of all *ε-subgradients* of f at x^0 is called the *ε-subdifferential* of f at x^0 and is denoted by $\partial_\varepsilon f\left(x^0\right)$.
We also have notations

$$\operatorname{dom} \partial f = \{x \in X : \partial f(x) \neq \emptyset\} \text{ and range } \partial f(x) = \cup\{\partial f(x) : x \in \operatorname{dom} \partial f\}$$

Proposition 1. *Let f be a proper convex function. Then*

1. *$\partial_\varepsilon f(x)$ is a closed convex set, for any $x \in X$ and $\varepsilon \geq 0$.*
2. *$\operatorname{ri}(\operatorname{dom} f) \subset \operatorname{dom} \partial f \subset \operatorname{dom} f$ where $\operatorname{ri}(\operatorname{dom} f)$ stands for the relative interior of $\operatorname{dom} f$.*
3. *If f has a unique subgradient at x, then f is differentiable at x, and $\partial f(x) = \{\nabla f(x)\}$.*
4. *$x_0 \in \operatorname{argmin}\{f(x) : x \in X\}$ if and only if $0 \in \partial f\left(x_0\right)$.*

Let C be a nonempty closed subset of $\mathbb{R}^n$. The indicator function $\chi_C(x) = 0$ if $x \in C, +\infty$ otherwise. For a closed subset C of $\mathbb{R}^n$, the normal cone of C, denoted by $N(C, x)$, is given by

$$N(C, x) = \partial \chi_C(x) = \left\{u \in \mathbb{R}^n : \langle u, y - x \rangle \leq 0 \quad \forall y \in C\right\}.$$

The function f is said to be λ-Lipschitz if

$$\|f\left(x_1\right) - f\left(x_2\right)\| \leq \lambda \left\|x_1 - x_2\right\| \quad \forall x_1, x_2 \in S.$$

The function f is called locally Lipshitz if for every $x \in S$ there exists a neiborhood U_x of x such that the restriction of f to U_x is Lipschitz.
Suppose that f is a locally Lipschitz function at a given $x \in \mathbb{R}^n$. The *Clark direction derivative* and the *Clark subdifferential* of f at x is given by the following formulas, respectively.

$$f^{\uparrow}(x, v) = \limsup_{(t,y)\to(0^+,x)} \frac{f(y + tv) - f(y)}{t},$$
$$\partial^{\uparrow} f(x) = \left\{x^* \in \mathbb{R}^n : \langle x^*, v \rangle \leq f^{\uparrow}(x, v) \quad \forall v \in \mathbb{R}^n\right\}.$$

If f is continuously differentiable at x then $\partial^{\uparrow} f(x) = \nabla f(x)$. When f is a convex function, then $\partial^{\uparrow} f(x)$ coincides with the subdifferential $\partial f(x)$.

Conjugates of convex functions

The *conjugate* of a function $f : X \mapsto \overline{\mathbb{R}}$ is the function $f^* : X \mapsto \overline{\mathbb{R}}$ defined by

$$f^*(y) = \sup_{x \in X} \{\langle x, y\rangle - f(x)\}$$

Proposition 2. *Let $f \in \Gamma_0(X)$. Then we have*

1. *$f^* \in \Gamma_0(X)$ and $f^{**} = f$.*
2. *$f(x) + f^*(y) \geq \langle x, y\rangle$, for any $x, y \in X$.*
   Equality holds if and only if $y \in \partial f(x) \Leftrightarrow x \in \partial f^*(y)$.
3. *$y \in \partial_\varepsilon f(x) \Longleftrightarrow x \in \partial_\varepsilon f^*(y) \Longleftrightarrow f(x) + f^*(y) \leq \langle x, y\rangle + \varepsilon$, for all $\varepsilon > 0$.*

Difference of convex (DC) functions

A function f is called DC function on X if it has the form

$$f(x) = g(x) - h(x), \quad x \in X$$

where g and h belong to $\Gamma_0(X)$. One says that $g - h$ is a *DC decomposition* of f and g, h are its *DC components*. If g and h are in addition finite on all of X then one says that $f = g - h$ is a finite DC function on X. The set of DC functions (resp. finite DC functions) on X is denoted by $\mathcal{DC}(X)$ (resp. $\mathcal{DC}_f(X)$).

Remark 1. *Given a DC function f with a DC decomposition $f = g - h$. Then for every $\theta \in \Gamma_0(X)$ finite on the whole X, $f = (g + \theta) - (h + \theta)$ is another DC decomposition of f. Thus, a DC function f has infinitely many DC decompositions.*

1.3.2 DC Programming

In this section, we provide a brief summary of the standard DC optimization's important results, which serve as the theoretical foundation for our methodologies.

Standard DC program

In the sequel, we use the convention $+\infty - (+\infty) = +\infty$.
For $g, h \in \Gamma_0(X)$, a standard DC program is that of the form

$$(P) \quad \alpha = \inf\{f(x) = g(x) - h(x) : x \in X\}$$

and its dual counterpart

$$(D) \quad \alpha^* = \inf \{h^*(y) - g^*(y) : y \in X\}$$

There is a perfect symmetry between primal and dual programs (P) and (D) : the dual program to (D) is exactly (P), moreover, $\alpha = \alpha^*$.

Remark 2. *Let C be a nonempty closed convex set. Then, the constrained problem*

$$\inf\{f(x) = g(x) - h(x) : x \in C\}$$

can be transformed into an unconstrained DC program by using the indicator function χ_C, i.e.,

$$\inf\{f(x) = \phi(x) - h(x) : x \in X\}$$

where $\phi := g + \chi_C$ is in $\Gamma_0(X)$.

We will consistently maintain the following assumption, which is derived from the finiteness of α.

$$\operatorname{dom} g \subset \operatorname{dom} h \text{ and } \operatorname{dom} h^* \subset \operatorname{dom} g^*.$$

Optimality conditions for standard DC optimization

A point x^* is said to be a local minimizer of $g - h$ if $x^* \in \operatorname{dom} g \cap \operatorname{dom} h$ (so, $(g - h)(x^*)$ is finite) and there is a neighborhood U of x^* such that

$$g(x) - h(x) \geq g(x^*) - h(x^*), \quad \forall x \in U.$$

A point x^* is said to be a critical point of $g - h$ if it verifies the generalized Kuhn-Tucker condition

$$\partial g(x^*) \cap \partial h(x^*) \neq \emptyset$$

Let $\mathcal{P}$ and $\mathcal{D}$ denote the solution sets of problems (P) and (D) respectively, and let

$$\mathcal{P}_\ell = \{x^* \in X : \partial h(x^*) \subset \partial g(x^*)\}, \quad \mathcal{D}_\ell = \{y^* \in X : \partial g^*(y^*) \subset \partial h^*(y^*)\}$$

We present some fundamental results on DC programming [74] below.

Theorem 1.

1. *Global optimality condition: $x \in \mathcal{P}$ if and only if*

$$\partial_\varepsilon h(x) \subset \partial_\varepsilon g(x), \forall \varepsilon > 0.$$

2. *Transportation of global minimizers:* $\cup\{\partial h(x) : x \in \mathcal{P}\} \subset \mathcal{D} \subset \operatorname{dom} h^*$. *The first inclusion becomes equality if* g^* *is subdifferentiable in* $\mathcal{D}$. *In this case* $\mathcal{D} \subset (\operatorname{dom} \partial g^* \cap \operatorname{dom} \partial h^*)$.

3. *Necessary local optimality: if* x^* *is a local minimizer of* $g - h$, *then* $x^* \in \mathcal{P}_\ell$.

4. *Sufficient local optimality: Let* x^* *be a critical point of* $g - h$ *and* $y^* \in \partial g(x^*) \cap \partial h(x^*)$. *Let* U *be a neighborhood of* x^* *such that* $(U \cap \operatorname{dom} g) \subset \operatorname{dom} \partial h$. *If for any* $x \in U \cap \operatorname{dom} g$, *there is* $y \in \partial h(x)$ *such that* $h^*(y) - g^*(y) \geq h^*(y^*) - g^*(y^*)$, *then* x^* *is a local minimizer of* $g - h$. *More precisely,*

$$g(x) - h(x) \geq g(x^*) - h(x^*), \quad \forall x \in U \cap \operatorname{dom} g$$

5. *Transportation of local minimizers: Let* $x^* \in \operatorname{dom} \partial h$ *be a local minimizer of* $g - h$. *Let* $y^* \in \partial h(x^*)$ *and a neighborhood* U *of* x^* *such that* $g(x) - h(x) \geq g(x^*) - h(x^*), \forall x \in U \cap \operatorname{dom} g$. *If*

$$y^* \in \operatorname{int}(\operatorname{dom} g^*) \text{ and } \partial g^*(y^*) \subset U$$

then y^* *is a local minimizer of* $h^* - g^*$.

Remark 3.

1. *By the symmetry of the DC duality, these results have their corresponding dual part. For example, if* y *is a local minimizer of* $h^* - g^*$, *then* $y \in \mathcal{D}_\ell$.

2. *The properties ii), iv) and their dual parts indicate that there is no gap between the problems* (P) *and* (D). *They show that globally/locally solving the primal problem* (P) *implies globally/locally solving the dual problem* (D) *and vice-versa. Thus, it is useful if one of them is easier to solve than the other.*

3. *The necessary local optimality condition* $\partial h^*(x^*) \subset \partial g^*(x^*)$ *is also sufficient for many important classes of programs, for example [51], if* h *is polyhedral convex, or when* f *is locally convex at* x^*, *i.e. there exists a convex neighborhood* U *of* x^* *such that* f *is finite and convex on* U. *We know that a polyhedral convex function is almost everywhere differentiable, that is it is differentiable everywhere except on a set of measure zero. Thus, if* h *is a polyhedral convex function, then a critical point of* $g - h$ *is almost always a local solution to* (P).

4. *If* f *is actually convex on* X, *we call* (P) *a "false" DC program. In addition, if* $\operatorname{ri}(\operatorname{dom} g) \cap \operatorname{ri}(\operatorname{dom} h) \neq \emptyset$ *and* $x^0 \in \operatorname{dom} g$ *such that* g *is continuous at* x^0, *then* $0 \in \partial f(x^0) \Leftrightarrow \partial h(x^0) \subset \partial g(x^0)$ *[51]. Thus, in this case, the local optimality is also sufficient for the global optimality. Consequently, if in addition* h *is differentiable, a critical point is also a global solution.*

1.3.3 Standard DC Algorithm (DCA)

The DCA consists in the construction of the two sequences $\left\{x^k\right\}$ and $\left\{y^k\right\}$ (candidates for being primal and dual solutions, respectively) which are easy to calculate and satisfy the following properties:

1. The sequences $(g-h)\left(x^k\right)$ and $(h^*-g^*)\left(y^k\right)$ are decreasing.
2. Their corresponding limits x^∞ and y^∞ satisfy the local optimality condition $(x^\infty, y^\infty) \in \mathcal{P}_\ell \times \mathcal{D}_\ell$ or are critical points of $g-h$ and h^*-g^*, respectively.

From a given point $x^0 \in \operatorname{dom} g$, DCA generates these sequences by the scheme

$$y^k \in \partial h\left(x^k\right) = \arg\min\left\{h^*(y) - \left\langle y, x^k\right\rangle : y \in X\right\} \tag{1.1}$$

$$x^{k+1} \in \partial g^*\left(y^k\right) = \arg\min\left\{g(x) - \left\langle x, y^k\right\rangle : x \in X\right\}. \tag{1.2}$$

The interpretation of the above scheme is simple. At iteration k of DCA, we replace the second component h in the primal DC program by its affine minorant

$$h_k(x) = h\left(x^k\right) + \left\langle x - x^k, y^k\right\rangle, \tag{1.3}$$

where $y^k \in \partial h\left(x^k\right)$. Then the original DC program reduces to the convex program

$$(P_k) \quad \alpha_k = \inf\left\{f_k(x) := g(x) - h_k(x) : x \in X\right\} \tag{1.4}$$

that is equivalent to 1.1. It is easy to see that f_k is a majorant of f at x^k. Similarly, by replacing g^* with its affine minorant

$$g_k^*(y) = g^*\left(y^{k-1}\right) + \left\langle y - y^{k-1}, x^k\right\rangle, \tag{1.5}$$

where $x^k \in \partial g^*\left(y^{k-1}\right)$, we lead to the convex problem

$$(D_k) \quad \inf\left\{h^*(y) - g_k^*(y) : y \in X\right\} \tag{1.6}$$

whose solution set is $\partial h\left(x^k\right)$.

Remark 4.

1. *Finding y^k, x^{k+1} by the scheme 1.4 is equivalent to solving the problems (D_k) and (P_k). Thus, DCA works by reducing a DC program to a sequence of convex programs that can be solved efficiently.*

2. *In practice, the calculation of the subgradient of the function h at a point x is usually easy if we know its explicit expression. But, the explicit expression of the conjugate of a given function g is unknown, so calculating x^{k+1} is done by solving the convex problem (P_k).*

3. *DCA is constructed from DC convex components g and h and their conjugates but not from the DC function f itself, while a DC function has finitely many DC decompositions. Thus, it is useful to find a suitable DC decomposition since it may have crucial impacts on the efficiency of DCA.*

Well definiteness of DCA

DCA is well defined if one can construct two sequences $\left\{x^k\right\}$ and $\left\{y^k\right\}$ as above from an arbitrary initial point x^0. The following lemma is the necessary and sufficient condition for this property

Lemma 1. *([74]) The sequences $\left\{x^k\right\}$ and $\left\{y^k\right\}$ in DCA are well defined if and only if*

$$\operatorname{dom}\partial g \subset \operatorname{dom}\partial h \text{ and } \operatorname{dom}\partial h^* \subset \operatorname{dom}\partial g^*$$

Since for $\varphi \in \Gamma_0(X)$ we have $\operatorname{ri}(\operatorname{dom}\varphi) \subset \operatorname{dom}\partial\varphi \subset \operatorname{dom}\varphi$ (Proposition 1.1). Moreover, we also keep the assumptions $\operatorname{dom} g \subset \operatorname{dom} h$, $\operatorname{dom} h^* \subset \operatorname{dom} g^*$. So, we can say that DCA in general is well defined.

Convergence properties of standard DCA

Complete convergence of standard DCA is given in the following results [73, 44, 44].

Theorem 2. *Suppose that the sequences $\left\{x^k\right\}$ and $\left\{y^k\right\}$ are generated by the DCA. Then we have*

1. *The sequences $\left\{g\left(x^k\right) - h\left(x^k\right)\right\}$ and $\left\{h^*\left(y^k\right) - g^*\left(y^k\right)\right\}$ are decreasing and*

   - *$g\left(x^{k+1}\right) - h\left(x^{k+1}\right) = g\left(x^k\right) - h\left(x^k\right)$ if and only if $\left\{x^k, x^{k+1}\right\} \subset \partial g^*\left(y^k\right) \cap \partial h^*\left(y^k\right)$ and $\left[\rho(h) + \rho(g)\right]\left\|x^{k+1} - x^k\right\| = 0$.*
   - *$h^*\left(y^{k+1}\right) - g^*\left(y^{k+1}\right) = h^*\left(y^k\right) - g^*\left(y^k\right)$ if and only if $\left\{y^k, y^{k+1}\right\} \subset \partial g\left(x^k\right) \cap \partial h\left(x^k\right)$ and $\left[\rho\left(h^*\right) + \rho\left(g^*\right)\right]\left\|y^{k+1} - y^k\right\| = 0$.*

   DCA terminates at the k-th iteration if either of the above equalities holds.

2. *If $\rho(h) + \rho(g) > 0$ (resp. $\rho\left(h^*\right) + \rho\left(g^*\right) > 0$), then the sequences $\left\{\left\|x^{k+1} - x^k\right\|^2\right\}$ (resp. $\left\{\left\|y^{k+1} - y^k\right\|^2\right\}$) converge.*

3. *If the optimal value α is finite and the sequences $\{x^k\}$ and $\{y^k\}$ are bounded, then every limit point x^∞ (resp. y^∞) of the sequence $\{x^k\}$ (resp. $\{y^k\}$) is critical point of $g - h$ (resp. $h^* - g^*$).*

4. *DCA has a linear convergence for general DC program.*

The standard DCA scheme is presented in the following algorithm:

Algorithm Standard DCA scheme

Initialization: Let $x^0 \in \mathbb{R}^n$. Set $k = 0$.
repeat
1. Calculate $y^k \in \partial h\left(x^k\right)$.
2. Calculate $x^{k+1} \in \operatorname{argmin}\left\{g(x) - \left\langle x, y^k\right\rangle : x \in \mathbb{R}^n\right\}$.
3. $k \leftarrow k + 1$.

until convergence of $\{x^k\}$.

For decades, numerous DCA-based algorithms have been developed to efficiently solve a diverse number of large-scale problems in many application areas (see, e.g., [40, 77]), in particular [39, 41]). A deeper insight into DCA has been given in [43, 75].

Chapter 2

A block coordinate DCA approach for large-scale kernel SVM

Abstract. In this chapter, we introduce a new block coordinate descent approach based on DCA called BC-DCASVM for efficiently training large-scale kernel SVM models. The proposed method employs a unified scheme that is capable of handling almost all common losses in SVM. The key advantage of the block coordinate approach is that it divides the high-dimensional optimization variables into blocks. Variables are then updated one block at a time while holding other blocks fixed. This allows the objective value to decrease substantially at each step. Critically, it avoids overflowing memory, which enables efficient training on big datasets. Numerical experiments are conducted intensively, which shows the algorithm's merits in both accuracy and computational cost.

2.1 Introduction

2.1.1 Context and related work

Support vector machines (SVM) that were first proposed by Vapnik [11, 22, 93] have been shown to be a powerful machine learning method in the last two decades. Despite its advantageous properties, SVM is incapable of effectively dealing with large-scale datasets encountered in a variety of applications such as image classification, bioinformatics, or text classification, which are often in the millions in terms of data samples and features, rendering the computational and storage costs for training SVM prohibitively expensive or even infeasible.

The content presented within this chapter has been extracted from the subsequent scholarly publication:
[1] Pham, V.T., Luu, H.P.H., Le Thi, H.A. (2022). A Block Coordinate DCA Approach for Large-Scale Kernel SVM. In: Nguyen, N.T., Manolopoulos, Y., Chbeir, R., Kozierkiewicz, A., Trawiński, B. (eds) Computational Collective Intelligence. ICCCI 2022. Lecture Notes in Computer Science(), vol 13501. Springer, Cham. https://doi.org/10.1007/978-3-031-16014-1_27.

The block coordinate (BC) approach has been widely studied to address this challenge. Thanks to their low iteration costs, low memory requirements, and the possibility of being implemented in parallel, block coordinate algorithms have evolved as essential tools for tackling some of the most challenging large-scale optimization issues. In principle, at each iteration, these methods choose a block of coordinates of the optimization variable to update while keeping the other variables fixed. As a result, the computation time is reduced while preventing the computer's memory from overflowing, which is crucial for big data computing.

Related Works. Various block coordinate algorithms for machine learning, including SVM have been investigated in recent years. Hsieh et al. [33] introduced a dual coordinate descent approach for linear SVM with L1 and L2-loss functions. Chou et al. [21] proposed several coordinate descent approaches for linear one-class SVM and SVDD. Nutini et al. [67] proposed greedy block-selection strategies to make block coordinate faster on some classic machine learning problems. Numerous large-scale optimization problems have been successfully solved using block-coordinate approaches and their modifications ([47, 52, 79]).

2.1.2 Motivation and contributions

Zhou and Zhou [111] proposed a unified SVM scheme using LS-DC loss for dealing with both the convex and nonconvex loss functions based on the difference of convex functions algorithms (DCA) [74, 75]. This approach is promising since most loss functions of SVM are LS-DC or can be approximated by LS-DC. However, to address the large-scale problem, they employed low-rank approximation, which might be inefficient in some circumstances when dealing with the kernel matrix in SVM. For example, in natural language processing, image processing, or bioinformatics, etc., where data have a large number of attributes, there is little chance that the data will be strongly correlated. Due to the weak correlation in these cases, low-rank approximation algorithms are unable to efficiently reduce the rank of the kernel matrix. Additionally, it is possible to lose a considerable amount of information if the training data are not strongly correlated. This motivates us to use another approach for solving the large-scale problem. That is, instead of reducing the problem's size by approximation methods, we opt for customizing the DCA scheme in such a way that it can work in a large-scale context. Specifically, in this chapter we focus on applying the block coordinate technique to DCA in order to address the large-scale kernel SVM.

Our contributions.

1. We propose a new block coordinate DCA based method for solving high-dimensional DC programs. Then, the proposed method is specifically applied to the large-scale kernel SVM in a unified manner that is capable of handling almost all common losses used in SVM, including convex and nonconvex losses, as well as those for classification and

regression. From the theoretical perspective, the proposed algorithm guarantees to find a weakly DC critical point (the precise definition is given later).

2. To further reduce the computational cost, we replace some large sums arising when applying the block coordinate DCA to the kernel SVM with their approximations based on the minibatch.

3. Finally, numerical experiments are carried out intensively for various datasets and loss functions. It is consistent that our proposed method outperforms existing algorithms in both solutions' quality and computational time.

2.2 Optimization problem of kernel SVM

Given a training dataset $\{(x_i, y_i)\}_{i=1}^m$ where x_i is a feature vector and y_i is the corresponding label, the kernel SVM that is based on structural risk minimization takes the following form

$$\min_{w\in\mathbb{H}} \frac{1}{m}\sum_{i=1}^{m} \ell(y_i, \langle w, \phi(x_i)\rangle) + \frac{\lambda}{2}\|w\|^2, \tag{2.1}$$

where $\mathbb{H}$ is a reproducible kernel Hilbert space induced by a kernel $\kappa(x, z) = \langle \phi(x), \phi(z)\rangle$ where ϕ is a feature map from $\mathbb{R}^d$ to $\mathbb{H}$, ℓ is a loss function measuring the fidelity of the prediction and the true label, $\lambda > 0$ is the regularization parameter. Once learned, the found w^* specifies a classifier $f(x) = \langle w^*, \phi(x)\rangle$ that is expected to separate two classes. Due to the high or even infinite dimension of $\mathbb{H}$, the problem (2.1) can not be solved efficiently. By applying the representor theorem [86, 87], we can substitute $w = \sum_{i=1}^m \boldsymbol{\alpha}_i \phi(x_i)$ in (2.1) to have the finite dimensional optimization problem as follows:

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^m} F(\boldsymbol{\alpha}) = \lambda \boldsymbol{\alpha}^\top \boldsymbol{K}\boldsymbol{\alpha} + \frac{1}{m}\sum_{i=1}^{m} \ell\left(y_i, \boldsymbol{K}_i\boldsymbol{\alpha}\right) \tag{2.2}$$

where $\boldsymbol{K}$ is the kernel matrix and $\boldsymbol{K_i}$ is the i-th row of $\boldsymbol{K}$.

The problem (2.2) is an optimization problem in $\mathbb{R}^m$ space, whose properties (convex, nonconvex) depend on the structure of ℓ. For regression tasks, ℓ is usually of the form $\ell(y, t) = \psi(y - t)$, where ψ can be the least square loss, absolute loss, ϵ-insensitive loss, etc. For classification tasks, ℓ is usually given as $\ell(y, t) = \psi(1 - yt)$ where ψ can be the least square loss, hinge loss, ramp loss, etc. Each choice of ℓ leads to a different SVM problem for which an algorithm must be designed. This is extremely inconvenient when there are numerous loss functions, with each function having different merits. Therefore, Zhou and Zhou suggested a unified model to simultaneously enjoy the benefits of various losses while minimizing effort in designing algorithms, namely the least squares type DC loss (LS-DC). It is pointed out in [111] that most losses used in the literature are LS-DC or can be approximated by LS-DC losses.

With ℓ being an LS-DC loss, the problem (2.2) is a DC program. However, the dimension of the optimization variable equals the number of data samples, posing a real scalability challenge.

2.3 Block coordinate DCA based method

2.3.1 Block coordinate DCA algorithm

When dealing with large-scale problems, data appear with an enormous number of observations and/or features, which eventually results in high-dimensional optimization problems. Consequently, the standard DCA could be prohibitively expensive or even impossible. This motivates us to design and analyze a block coordinate DCA scheme to tackle a (high dimensional) DC program (P_{dc}) in general. Let us denote $\Omega = \{A : A \subset \{1, 2, \ldots, m\}, A \neq \emptyset\}$, which is the set of all non-empty subsets of $\{1, 2, \ldots, m\}$. A block of coordinates corresponds to an element of Ω. Now we prescribe a probability distribution $\mathcal{P}$ over Ω. This probability distribution represents a strategy on how likely a block is chosen at each iteration. In a certain strategy, one can simply assign a probability of 0 to blocks that are not supposed to be chosen. With a prescribed probability distribution over Ω, at each iteration, we randomly generate a set S with respect to the probability distribution $\mathcal{P}$ and perform one step of the DCA on the set S of coordinates. The formal procedure is described in Algorithm 1.

Algorithm 1 Block coordinate DCA

Input: A starting point $\boldsymbol{\alpha}^0$, a distribution $\mathcal{P}$ over Ω; Set $k = 0$.
repeat
1. Generate from the distribution $\mathcal{P}$ a set of coordinates S.
2. Compute a partial subgradient $\boldsymbol{\beta}^k \in \partial_{\boldsymbol{\alpha}_S} H(\boldsymbol{\alpha}_S^k, \boldsymbol{\alpha}_{S^C}^k)$.
3. Solve the convex problem: $\boldsymbol{\alpha}_S^* \in \operatorname{argmin}_{\boldsymbol{\alpha}_S} \{G(\boldsymbol{\alpha}_S, \boldsymbol{\alpha}_{S^C}^k) - \langle \boldsymbol{\beta}^k, \boldsymbol{\alpha}_S \rangle\}$.
4. Update solution $\boldsymbol{\alpha}_i^{k+1} = \begin{cases} \boldsymbol{\alpha}_i^* & \text{if } i \in S, \\ \boldsymbol{\alpha}_i^k & \text{otherwise.} \end{cases}$
5. Set $k \leftarrow k + 1$.

until Stopping criterion.

For a given distribution $\mathcal{P}$, we denote $\operatorname{Supp}(\mathcal{P}) = \{A \in \Omega : \mathcal{P}(A) > 0\}$ the support of $\mathcal{P}$. We have established the following convergence results.

Proposition 1. *Assume that either G or H is strongly convex. Then, almost surely, every limit point $\boldsymbol{\alpha}^*$ of the sequence $\{\boldsymbol{\alpha}^k\}$ generated by the block coordinate DCA satisfies: $\forall S \in \operatorname{Supp}(\mathcal{P}) : \partial_{\boldsymbol{\alpha}_S} G(\boldsymbol{\alpha}^*) \cap \partial_{\boldsymbol{\alpha}_S} H(\boldsymbol{\alpha}^*) \neq \emptyset$.*

In other words, the found solution $\boldsymbol{\alpha}^*$ is DC critical with respect to each block of coordinates that is assigned a non-zero probability. We call such a point weakly DC critical point with respect to the mentioned blocks of coordinates.

2.3.2 Application to kernel SVM with LS-DC losses

According to Zhou and Zhou [111], a loss ℓ is called an LS-DC loss if there exists a constant $A > 0$ such that the associated function ψ, i.e. $\ell(y,t) = \psi(1-yt)$, has the following DC decomposition $\psi(u) = Au^2 - (Au^2 - \psi(u))$.

Let $\ell(y,t)$ be any LS - DC loss associated with ψ, the kernel SVM model (2.2) can be written as the following DC program

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^m} F(\boldsymbol{\alpha}) = \lambda\boldsymbol{\alpha}^\top \boldsymbol{K}\boldsymbol{\alpha} + \frac{1}{m}\sum_{i=1}^{m}\psi\left(1 - y_i\boldsymbol{K}_i\boldsymbol{\alpha}\right) := G(\boldsymbol{\alpha}) - H(\boldsymbol{\alpha}) \tag{2.3}$$

with the DC components being given by

$$G(\boldsymbol{\alpha}) := \lambda\boldsymbol{\alpha}^\top \boldsymbol{K}\boldsymbol{\alpha} + \frac{1}{m}A\sum_{i=1}^{m}(1 - y_i\boldsymbol{K}_i\boldsymbol{\alpha})^2 + \frac{\rho}{2}\|\boldsymbol{\alpha}\|^2, \tag{2.4}$$

$$H(\boldsymbol{\alpha}) := \frac{1}{m}\sum_{i=1}^{m}\left(A(1 - y_i\boldsymbol{K}_i\boldsymbol{\alpha})^2 - \psi\left(1 - y_i\boldsymbol{K}_i\boldsymbol{\alpha}\right)\right) + \frac{\rho}{2}\|\boldsymbol{\alpha}\|^2, \tag{2.5}$$

where $\rho > 0$ is a regularization parameter to make both DC components strongly convex.

Since the DC program (3.4) is high dimensional, we shall employ the Block coordinate DCA to handle it. At iteration k, we choose a block of coordinates $S \in \mathcal{D} := \{1, 2, \ldots, m\}$ ($1 \leq |S| < m$). Then, the partial subgradient of H on the set of coordinates S, $\beta \in \partial_{\alpha_S} H(\boldsymbol{\alpha}^k_S, \boldsymbol{\alpha}^k_{Sc})$, is computed as

$$\begin{aligned}\boldsymbol{\beta}^k = &\frac{1}{m}\sum_{i=1}^{m} 2A(1 - y_i\boldsymbol{K}_i\boldsymbol{\alpha}^{\boldsymbol{k}})(-y_i\boldsymbol{K}_i^{[S]\top}) \\ &- \frac{1}{m}\sum_{i=1}^{m}\left(\psi'\left(1 - y_i\boldsymbol{K}_i\boldsymbol{\alpha}^{\boldsymbol{k}}\right)(-y_i\boldsymbol{K}_i^{[S]\top})\right) + \rho\boldsymbol{\alpha}^k_S,\end{aligned}$$

where $\psi'(u)$ denotes a subgradient of ψ at u. To further reduce the computational cost, we approximate the two large sums above (each with m components) by a minibatch of data. For convenience, the minibatch index set is chosen to be the same as S. So $\boldsymbol{\beta}^k$ is approximated by:

$$\begin{aligned}\tilde{\boldsymbol{\beta}}^k = &\frac{1}{|S|}\sum_{i\in S} 2A(1 - y_i\boldsymbol{K}_i\boldsymbol{\alpha}^{\boldsymbol{k}})(-y_i\boldsymbol{K}_i^{[S]\top}) \\ &- \frac{1}{|S|}\sum_{i\in S}\left(\psi'\left(1 - y_i\boldsymbol{K}_i\boldsymbol{\alpha}^{\boldsymbol{k}}\right)(-y_i\boldsymbol{K}_i^{[S]\top})\right) + \rho\boldsymbol{\alpha}^k_S.\end{aligned} \tag{2.6}$$

Then, we solve the following convex subproblem

$$\min_{\boldsymbol{\alpha}_S} G(\boldsymbol{\alpha}_S, \boldsymbol{\alpha}_{SC}^k) - \langle \tilde{\boldsymbol{\beta}}^k, \boldsymbol{\alpha}_S \rangle. \tag{2.7}$$

We write the problem (2.7) in detail as follows. The first term of $G(\boldsymbol{\alpha})$ in (2.4) can be expanded:

$$[\boldsymbol{\alpha}_S, \boldsymbol{\alpha}_S^k c]^\top \boldsymbol{K} [\boldsymbol{\alpha}_S, \boldsymbol{\alpha}_{Sc}^k] = \boldsymbol{\alpha_S}^\top \boldsymbol{K}_{SS} \boldsymbol{\alpha}_S + 2\boldsymbol{\alpha}_S^\top \boldsymbol{K}_{SSc} \boldsymbol{\alpha}_{Sc}^k + \boldsymbol{\alpha}_{Sc}^\top \boldsymbol{K}_{ScSc} \boldsymbol{\alpha}_{Sc}^k \tag{2.8}$$

The second term of $G(\boldsymbol{\alpha})$ is as follows:

$$-2 \left[\sum_{i=1}^m y_i \boldsymbol{K}_i^S \right] \boldsymbol{\alpha}_S + \boldsymbol{\alpha}_S^\top \left[\sum_{i=1}^m y_i^2 (\boldsymbol{K}_i^S)^\top \boldsymbol{K}_i^S \right] \boldsymbol{\alpha}_S + 2\boldsymbol{\alpha}_S^\top \left[\sum_{i=1}^m y_i^2 (K_i^S)^\top K_i^{Sc} \right] \boldsymbol{\alpha}_{Sc}^k + C \tag{2.9}$$

where C is a constant which does not depend on $\boldsymbol{\alpha}$.

The optimization problem (2.7) can be written as follows:

$$\begin{aligned}
&\min_{\alpha_S} \lambda \left(\boldsymbol{\alpha_S}^\top \boldsymbol{K}_{SS} \boldsymbol{\alpha}_S + 2\boldsymbol{\alpha}_S^\top \boldsymbol{K}_{SSc} \boldsymbol{\alpha}_{Sc}^k \right) \\
&+ \frac{A}{|S|} \left(-2 (\sum_{i \in S} y_i K_i^S) \boldsymbol{\alpha}_S + \boldsymbol{\alpha}_S^\top (\sum_{i \in S} y_i^2 (K_i^S)^\top K_i^S) \boldsymbol{\alpha}_S + 2\boldsymbol{\alpha}_S^\top (\sum_{i \in S} y_i^2 (K_i^S)^\top K_i^{Sc}) \boldsymbol{\alpha}_{Sc}^k \right) \\
&+ \frac{\rho}{2} \|\boldsymbol{\alpha}_S\|^2 - \langle \boldsymbol{\beta}^k, \boldsymbol{\alpha}_S \rangle \\
&= \min_{\alpha_S} \boldsymbol{\alpha_S}^\top \left(\lambda \boldsymbol{K}_{SS} + \frac{A}{|S|} \sum_{i \in S} y_i^2 (K_i^S)^\top K_i^S + \frac{\rho}{2} I \right) \boldsymbol{\alpha}_S \\
&+ \boldsymbol{\alpha}_S^\top \left(2\lambda K_{SSc} \boldsymbol{\alpha}_{Sc}^k - \frac{2A}{|S|} \sum_{i \in S} y_i (K_i^S)^\top + \frac{2A}{|S|} (\sum_{i \in S} y_i^2 (K_i^S)^\top K_i^{Sc}) \boldsymbol{\alpha}_{Sc}^k - \boldsymbol{\beta}^k \right).
\end{aligned} \tag{2.10}$$

This is a positive definite quadratic program, which results in a system of linear equations being solved at each iteration. Because $\boldsymbol{\alpha}_S$ has a relatively small dimension, the solution can be efficiently solved using existing packages.

2.3.3 Block selection rule

Given a set of possible blocks, the block selection rule is used to identify which block of the optimization variable to update. In our proposed method, we examined the random strategy for selecting a block at each iteration. On the other hand, Lee and Wright [48] provided an analysis to show that the random permutations cyclic selection (RPCD) is slightly better than the random strategy (RCD) in terms of convergence rate. Therefore, we consider two strategies

for selecting a block of variables S at each iteration: Random selection and cyclic selection with random permutations. For the random strategy, the term *iteration* refers to the number of times a block of variables passes through the algorithm. For the cyclic strategy, the number of times the algorithm will run over the full training dataset is represented by the term *epoch*.

2.3.4 Block coordinate DCA for kernel SVM algorithm

Our proposed algorithm: Block coordinate DCA for kernel SVM algorithm is referred to as BC-DCASVM, and is presented in Algorithm 2.

Algorithm 2 BC-DCASVM

Input: Given a training set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$ with $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$; Kernel matrix $\boldsymbol{K}$ satisfying $\boldsymbol{K}_{i,j} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$; Any LS-DC loss function $\psi(u)$ with parameter $A > 0$; Set $e = 0$, $\boldsymbol{\alpha} = \mathbf{0}$, $M > 0$

repeat

for $k = 1 \dots M$ **do**

1. Pick a block S cyclically or randomly.
2. Compute $\tilde{\boldsymbol{\beta}}^k$ in (2.6).
3. Solve (3.1) to obtain $\boldsymbol{\alpha}_S^*$.
4. Update $\boldsymbol{\alpha}^{k+1}$.

end for

5. Set $e \leftarrow e + 1$.

until Stopping criterion.

2.4 Datasets and numerical settings

In this section, we analyze the performance of our proposed algorithm BC-DCASVM with two state-of-the-art SVM algorithms, including UniSVM [111] and LibSVM [17] on some large datasets. To evaluate the two unified models: BC-DCSVM and UniSVM, four LS-DC loss functions (for both convex and nonconvex) and their corresponding subdiferentials will be used in our experiments:

1. Least squares: $\psi(u) = u^2$ is a convex loss function; $\nabla\psi(u) = 2u$.

2. Truncated-least-squares: $\psi(u) = \min\{u^2, a\}$, with $a > 0$ is a nonconvex function; $\partial\psi(u) = \begin{cases} 2u, & |u| < \sqrt{a} \\ 0, & |u| \geq \sqrt{a} \end{cases}$.

3. Squared Hinge: $\psi(u) = u_+^2$ with A ≥ 1 is a convex function; $\nabla\psi(u) = 2u_+$.

4. Truncated squared Hinge: $\psi(u) = \min\left\{u_+^2, a\right\}$ with $A \geq 1$ and the truncated parameter $a > 0$ is a nonconvex function; $\partial\psi(u) = \begin{cases} 2u, & 0 < u < \sqrt{a}, \\ 0, & \text{others} \end{cases}$.

For each $\psi(u)$, we replace a concrete subgradient as given above to (2.6) to get the correspond-

ing algorithm.

2.4.1 Datasets

We perform the related algorithms on ten well-known benchmark datasets: Five datasets [NEWS20, RCV1, GISETTE, MNIST, CIFA10] are from the LIBSVM website[3]. MNIST-17 (Dataset consists of labels: 1 and 7), MNIST-38 (Labels: 3 and 8), CIFA10-CP (Labels: Car and Plane) and CIFA10-TS (Labels: Truck and Ship) are the sub-datasets which extracted from two image datasets MNIST and CIFA10. DEXTER is a text classification dataset from UCI Machine Learning repository[4]. PCMAC dataset (PC vs Mac) is a subset of the 20 Newsgroups[5] used in [109]. GLI-85 is a microarray-bio dataset[6]. Table 2.1 summarizes the information included in the datasets. Additional information of these datasets is available on the LIBSVM website and the UCI Machine Learning Repository.

dataset	name	#samples	#features
D1	GLI-85	85	22,283
D2	DEXTER	600	20,000
D3	PCMAC	1,943	3,289
D4	GISETTE	7,000	5,000
D5	MNIST-17	15,170	784
D6	MNIST-38	13,728	784
D7	CIFA10-TS	12,000	3,072
D8	CIFA10-CP	12,000	3,072
D9	RCV1	20,242	47,236
D10	NEWS20	19,996	1,355,191

Table 2.1: Information of the datasets (samples (m), features (n)) used in our experiments.

For model selection purposes, we use the 5-fold cross-validation strategy to search for the best parameters (grid search). The binary datasets have the labels in $\{+1, -1\}$ will be split into two subsets of the labels $[+1]$ and $[-1]$. To avoid overfitting and preserve the proportions of the classes, each subset will be divided into 5 equal-sized subfolds, and then the subfolds of the label $[+1]$ will be merged with the corresponding subfolds of the label $[-1]$ to produce 5 folds for training and testing. For evaluation, each fold will serve in turn as the testset, while the remaining four folds will serve as the trainset. Before feeding the training data into the learning models, the whole data will be shuffled, and then scaled by the median absolute deviation in the range of $[0, 1]$ by using the function *normalize*[7] in MATLAB 2021a to avoid one feature having a greater influence than the others in the Gaussian kernel values. The same process is

[3] https://www.csie.ntu.edu.tw/∼cjlin/libsvm/
[4] https://archive.ics.uci.edu/
[5] http://qwone.com/∼jason/20newsgroups/
[6] https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=gse4412
[7] www.mathworks.com/help/releases/r2021a/matlab/ref/double.normalize.html

applied to the test data. Finally, mean accuracy is obtained by averaging the results. The best model is the one that achieves the highest average accuracy on the testset.

2.4.2 Set up experiments and parameters

The two algorithms BC-DCASVM and UniSVM are implemented in MATLAB. The source-code of UniSVM[8] is from the authors in [111]. The LibSVM model is compiled from the LibSVM C++ source-code[3] as MEX-files and run in MATLAB. All algorithms are performed on a PC with an Intel Core i7-8700 CPU @3.20GHz×6 with a memory of 16GB. The computer runs Windows 10 with the MATLAB-2021a version. For all algorithms, we particularly consider the Gaussian kernel: $\kappa(x, \boldsymbol{z}) = \exp(-\gamma\|x - \boldsymbol{z}\|^2)$ and used the following sets of candidate values: $\{2^{-10}, 2^{-8}, 2^{-6}, 2^{-4}, 2^{-2}, 2^0, 2^2\}$ and $\{2^{-2}, 2^0, 2^2, 2^4, 2^6, 2^8, 2^{10}\}$ respectively of the parameter γ (for the Gaussian kernel) and λ (the regularization parameter) in our experiments. For the proposed algorithm, $\boldsymbol{\alpha}_0$ is set as a vector with all zeros, ρ is set with value 10^{-5}. To solve the system of linear equations in (3.1), we use the function **linsolve**[9] in MATLAB. The block-sizes are set with different values depending on the number of samples of datasets as follows:

GLI-85	(32)	PCMAC	(128)	MNIST-38	(1024)	MNIST-17	(1024)	RCV1	(1024)
GISETTE	(128)	DEXTER	(128)	CIFA10-CP	(1024)	CIFA10-CP	(1024)	NEWS20	(1024)

2.5 Experiments and Discussion

We use three criteria for evaluating the performances of the comparative algorithms: the train accuracy (TRAIN: in %), the test accuracy (TEST: in %) and CPU runtime (CPU: in seconds). After tuning, we assess the models using the mean and the standard deviation of each evaluation criteria. The bold values in the two tables are the best results. For any algorithm that is unable to complete tasks due to the long training period (exceeding 1800 seconds), we denote the outcomes as N/A. Three experiments will be conducted to comprehensively analyze the proposed approach.

2.5.1 Experiment 1: Performance on large benchmark datasets

We evaluate the performances of BC-DCASVM, UniSVM and LibSVM on 10 datasets (as described in Table 2.1). The BC-DCASVM and UniSVM models are trained with the Least square loss function, which produced the best experimental results in terms of runtime, as Zhou and Zhou [111] stated in their experiment. For our proposed algorithm, testing indicates that just going over all of the data once is sufficient to get comparable results. Therefore, we

[8] https://github.com/stayones/code-unisvm
[9] www.mathworks.com/help/releases/r2021a/matlab/ref/linsolve.html

use a single epoch with the random permutations cyclic technique for BC-DCASVM in this experiment. The results of the experiment are in Table 2.2.

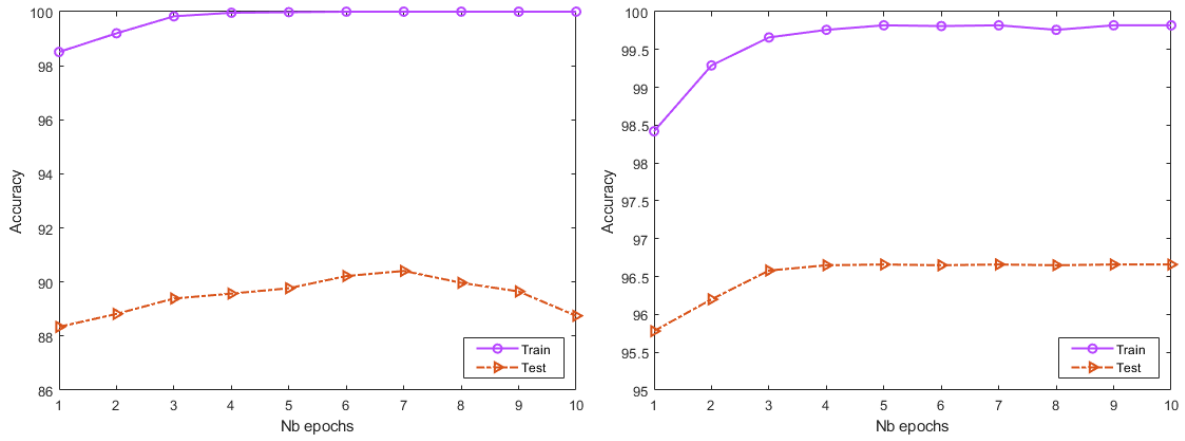
Dataset	Metric	BC-DCASVM	UniSVM	LibSVM
D1	TRAIN	**100(0.00)**	**100(0.00)**	**100(0.00)**
	TEST	**90.00(0.00)**	**90.00(0.00)**	83.75(0.08)
	CPU	**0.05(0.00)**	0.06(0.00)	0.19(0.00)
D2	TRAIN	**100(0.00)**	99.25(0.00)	**100(0.00)**
	TEST	93.89(0.01)	**94.00(0.01)**	**94.00(0.02)**
	CPU	**0.13(0.00)**	1.38(0.01)	0.33(0.01)
D3	TRAIN	**99.25(0.00)**	94.83(0.01)	96.10(0.01)
	TEST	**90.82(0.01)**	**90.82(0.02)**	88.87(0.01)
	CPU	**0.30(0.00)**	1.80(0.08)	1.27(0.02)
D4	TRAIN	**100(0.00)**	98.45(0.00)	**100(0.00)**
	TEST	**98.29(0.00)**	97.54(0.00)	97.96(0.01)
	CPU	**3.13(0.01)**	10.60(0.19)	176.60(1.37)
D5	TRAIN	**100(0.00)**	**100(0.00)**	**100(0.00)**
	TEST	**99.84(0.00)**	99.62(0.00)	99.75(0.01)
	CPU	**2.01(0.03)**	5.22(0.04)	196.47(2.42)
D6	TRAIN	**100(0.00)**	99.84(0.00)	**100(0.00)**
	TEST	**99.77(0.00)**	99.63(0.00)	99.75(0.00)
	CPU	**2.10(0.17)**	5.80(0.01)	165.48(2.03)
D7	TRAIN	**99.77(0.00)**	87.39(0.00)	**99.77(0.01)**
	TEST	**88.48(0.00)**	86.81(0.01)	88.28(0.02)
	CPU	**5.36(0.08)**	10.68(0.08)	762.84(4.85)
D8	TRAIN	98.51(0.00)	88.57(0.01)	**100(0.00)**
	TEST	88.34(0.00)	87.36(0.01)	**89.05(0.02)**
	CPU	**5.21(0.07)**	9.81(0.03)	685.26(5.01)
D9	TRAIN	99.38(0.02)	92.83(0.02)	**99.81(0.00)**
	TEST	**96.94(0.02)**	92.61(0.01)	96.65(0.00)
	CPU	**7.08(0.06)**	8.80(0.16)	1035.27(3.48)
D10	TRAIN	**98.45(0.00)**	84.02(0.00)	N/A
	TEST	**95.83(0.00)**	83.87(0.01)	N/A
	CPU	**18.04(0.14)**	39.12(0.11)	N/A

Table 2.2: Performances of the related algorithms on 10 benchmark datasets. All results are averaged over 5 folds with the standard deviations.

2.5.2 Experiment 2: Parameter analysis

This experiment is designed to validate our hypothesis on the proposed algorithm: The more training epochs, the higher accuracy of the model. We conduct the experiment with ten runs of the BC-DCASVM algorithm with numbers of epochs in $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ respectively on two datasets: CIFA10-CP and NEWS20. The optimal hyperparameters obtained from the model selection process in **Experiment 1** are used for training. Also, the Least square loss

function is utilized to assess the influence of increasing the number of epochs on accuracy. The cyclic strategy is used for block selection rule and the metrics: Accuracies on the trainset and on the testset are used as evaluation criteria. Fig. 2.1 illustrates the results of this experiment.



(a) CIFA10-CP (b) NEWS20

Figure 2.1: Performances of BC-DCASVM on CIFA10-CP (2.1(a)) and NEWS20 (2.1(b)) as the number of epochs increases.

Comments on the results of Experiment 1 and Experiment 2: According to Table 2.2, our approach outperforms the comparative algorithms on almost all benchmark datasets. BC-DCASVM is much faster than UniSVM, and greatly outperforms LibSVM in terms of runtime. In some cases, BC-DCASVM is 5 times faster than UniSVM (for example, on D2 and D3) and 100 times faster than LibSVM (on D7, D8, D9 and D10). As the dataset size increases, while the classic SVM becomes slower (or even fails to handle), and UniSVM appears to make a trade-off between accuracy and runtime, our method shows efficiency by working on smaller data blocks. It is also important to note that, even though our approach is trained with only one epoch, it outperforms the two state-of-the-art algorithms in terms of accuracy in most datasets. That is, if runtime is not critical and accuracy is the primary concern, the algorithm can be trained with more epochs to achieve higher accuracy. However, without contradicting machine learning theory, Fig. 2.1 demonstrates that if training takes an excessive number of epochs, the performance does not improve (even decreases) due to the model overfitting.

2.5.3 Experiment 3: Robustness on difference loss functions

To determine the effectiveness of the unified scheme, we compare the two unified algorithms BC-DCASVM and UniSVM using different loss functions. Three additional loss functions are investigated: One convex loss function-squared Hinge loss and two non-convex loss functions: Truncated least square loss (a = 2) and truncated squared Hinge loss (a = 2). Five datasets: D2, D4, D6, D8, and D10 are utilized for this experiment. Three metrics: Train accuracy, test accuracy, and runtime are evaluated on five datasets with the random selection strategy

Dataset	Metric	Trunc-Least-Square		Trunc-Squared-Hinge		Squared-Hinge	
		BC-DCASVM	UniSVM	BC-DCASVM	UniSVM	BC-DCASVM	UniSVM
D2	TRAIN	**100(0.00)**	**100(0.00)**	**100(0.00)**	**100(0.00)**	**100(0.00)**	**100(0.00)**
	TEST	**94.00(0.01)**	**94.00(0.01)**	94.67(0.01)	**94.83(0.00)**	**94.00(0.00)**	**94.00(0.00)**
	CPU	**0.20(0.02)**	1.28(0.03)	**0.18(0.01)**	1.13(0.01)	**0.24(0.01)**	1.34(0.01)
D4	TRAIN	**99.89(0.00)**	98.60(0.00)	**100(0.00)**	98.45(0.00)	**100(0.00)**	99.52(0.00)
	TEST	**97.81(0.00)**	97.27(0.00)	**98.21(0.00)**	97.50(0.00)	**98.06(0.00)**	98.00(0.00)
	CPU	**3.25(0.07)**	9.13(0.15)	**3.37(0.09)**	14.68(0.20)	**3.11(0.05)**	11.87(0.03)
D6	TRAIN	**100(0.00)**	99.84(0.00)	**100(0.00)**	99.95(0.00)	**100(0.00)**	99.98(0.00)
	TEST	**99.77(0.00)**	99.65(0.00)	**99.75(0.00)**	99.71(0.00)	**99.77(0.00)**	**99.77(0.00)**
	CPU	**2.04(0.02)**	5.42(0.02)	**2.01(0.07)**	5.33(0.09)	**2.01(0.02)**	6.22(0.06)
D8	TRAIN	**96.57(0.02)**	89.91(0.00)	**99.84(0.02)**	91.53(0.01)	**99.84(0.00)**	91.53(0.00)
	TEST	**87.22(0.03)**	87.06(0.01)	**89.36(0.03)**	87.84(0.02)	**89.36(0.00)**	87.84(0.00)
	CPU	**5.78(0.05)**	10.90(0.09)	**5.83(0.15)**	12.97(0.25)	**5.75(0.18)**	12.98(0.15)
D10	TRAIN	**99.70(0.00)**	87.28(0.01)	**99.58(0.00)**	92.93(0.00)	**99.58(0.00)**	93.98(0.01)
	TEST	**96.61(0.00)**	86.58(0.01)	**96.63(0.00)**	91.98(0.00)	**96.63(0.00)**	92.15(0.00)
	CPU	**18.97(0.08)**	39.73(1.37)	**18.08(0.15)**	45.37(0.64)	**18.33(0.17)**	41.56(0.34)

Table 2.3: Performances of the two unified algorithms: BC-DCASVM and UniSVM with three loss functions: Truncated least square (Trunc-Least-Square), truncated squared Hinge (Trunc-Squared-Hinge) and Squared Hinge.

(Number of iterations $M = \left\lceil \frac{m}{block\_size} \right\rceil$. Table 2.3 summarizes the experimental outcomes.

Comments on the result of Experiment 3: The latest outcome of the experimentation, as depicted in Table 2.3, demonstrates a noteworthy trend. By utilizing three additional loss functions, our approach showcases superior performance compared to UniSVM across a majority of evaluation metrics. Importantly, this enhanced performance is consistently maintained when compared to the results from Experiment 1. That means, our technique can be effectively applicable to a variety of LS-DC loss functions. This observation substantiates the robustness and reliability of our technique, underscoring its potential for effective application across a diverse spectrum of LS-DC loss functions.

2.6 Conclusion

To conclude, in this chapter, we have proposed a new approach for solving the large-scale kernel SVM, which applies DCA efficiently in combination with the block coordinate technique. Owing to the block coordinate approach with a nice DC decomposition, the proposed algorithm bc-DCASVM can efficiently handle datasets with high dimensionality and a large number of instances, while maintaining competitive performance, which is often a trade-off for existing large-scale algorithms. Thus, our method can overcome the storage and computation bottlenecks of classic SVM when dealing with big-data. Despite the noisy and partial updates of the block coordinate approach, our research shows that the power of DCA still remains. With the unified scheme, like UniSVM, BC-DCASVM can apply to most of the existing loss func-

tions. However, while UniSVM uses the low-rank method for approximation, which may be a limitation when dealing with non-strongly correlated datasets, our solution employs the block coordinate approach that is independent of data correlation. The experimental results on ten real-world datasets show that our algorithm is more effective than the two mentioned methods.

Nonetheless, the proposed method has the following shortcomings that will be addressed in our future research. First, there is a gap between theory and practice, as the approximation errors caused by the use of mini-batch gradient have not been quantified. In addition, when the overlap occurs in computing the kernel matrix over multiple epochs (iterations), our implementation has not been able to reuse the calculations from the previous iteration in the current iteration. Moreover, there are aspects of the proposed algorithm that need to be further studied, such as investigating better strategies for block selection rule and evaluating the effectiveness of nonconvex losses in handling noisy data.

Chapter 3

DCA-based bagging approaches and application to solve the financial imbalanced data

Abstract. Ensemble learning techniques such as bagging have shown promise in addressing issues with imbalanced classification. Bagging involves training multiple independent models on resampled data subsets and combining their predictions, improving overall performance. However, standard bagging does not adequately address high imbalance common to financial tasks involving rare but important events like fraudulent transactions or loan defaults. In this chapter, we propose and evaluate two weighted bagging methods, one for general cases and one tailored for financial imbalanced datasets. First, we introduce BaggingDCA, which assigns each base model a weight based on its performance thanks to LS-DC, a unified approach for various loss functions in ML, both convex and non-convex. We evaluate the BaggingDCA algorithm on several benchmark datasets and compared its performance to existing bagging methods. Furthermore, to directly mitigate imbalance, we develop the CSB-DCA method by incorporating the cost-sensitive approach to BaggingDCA. The weighted bagging method combines multiple base classifiers trained on different subsets of the training data with different weights, which helps to improve the overall performance of the classification model. In addition, the cost-sensitive approach takes into account the cost of misclassifying instances of the minority class, which is typically higher than that of the majority class. To assess the effectiveness of CSB-DCA, we perform thorough numerical experiments using some financial datasets from UCI machine learning repository, Kaggle competitions and our financial dataset.

The content presented within this chapter has been extracted from the subsequent scholarly publications:

[1] Pham, V.T., Le Thi, H.A., Luu, H.P.H., Damel, P. (2023). DCA-Based Weighted Bagging: A New Ensemble Learning Approach. In: Nguyen, N.T., et al. Intelligent Information and Database Systems. ACIIDS 2023. Lecture Notes in Computer Science(), vol 13996. Springer, Singapore. https://doi.org/10.1007/978-981-99-5837-5_11

[2] Pham, V.T., Le Thi, H.A. and Damel, P. (2023). Cost-sensitive weighted bagging DCA-based method for imbalanced financial data. **Submitted in**: *Proceedings of the 4th International Conference and Summer School on Numerical Computations: Theory and Algorithms NUMTA 2023*.

3.1 Introduction and related works

3.1.1 Ensemble learning and bagging

Ensemble learning is defined as a method for combining predictions from multiple individual ML models to produce a final prediction [14]. This is commonly referred to as “the wisdom of the crowd", which indicates that the collective intelligence of a group of individuals is often more reliable and accurate than the intelligence of any individual member. Any form of ML algorithm may be used as an ensemble learner (e.g., Decision trees, K-nearest neighbor, SVM, Neural network, etc.). Ensemble learning has been proven as an effective method in ML and successfully applied in a variety of domains, including image recognition, natural language processing, and financial prediction [58, 85, 97].

Bagging [13], which stands for bootstrap aggregating, is one of the most popular techniques in ensemble learning. Bagging is a well-studied subject in ML and the practice of bagging has been demonstrated to be an effective method in a variety of fields, including classification, regression, and anomaly detection. It involves aggregating the predictions from a variety of models that were trained on various subsets of the training data in order to obtain the final outcome. For the past few years, bagging has been the topic of extensive study, and various modifications and additions have been proposed to further improve its effectiveness. Although bagging has demonstrated outstanding performance and is widely used in many applications, it appears that some modifications can be made on how the base learners are combined to improve the bagging predictions [10, 62, 63, 97].

In ML, there are numerous loss functions because they are intended to optimize different objectives, provide different levels of robustness, interpretability, computing efficiency and are compatible with various regularization techniques. For instance, when the objective is to minimize the difference between the predicted output and the actual output, the least square loss is a valid choice. However, when the objective is to minimize the impact of noisy or mislabelled data points, the ramp loss is a better option. While having many loss functions to choose from can be advantageous in some cases, it is important to consider the potential downsides, such as added complexity in designing ML algorithms. Zhou and Zhou [111] defined a type of loss with a DC decomposition, named LS-DC loss, and demonstrated that all commonly used losses, both convex and nonconvex loss functions, as well as for both classification and regression problems, are LS-DC loss or can be approximated by LS-DC loss. As numerous loss functions can now be utilized with the same implementation and in the same ML algorithm, this approach makes the algorithm powerfully applicable and has a great deal of potential. Zhou and Zhou applied the concept of LS-DC to the SVM algorithm for solving the large-scale problem. In this study, we bring this method to the bagging technique in a new way that enhances the accuracy and

robustness of model prediction.

3.1.2 Financial problems

Financial data analysis plays a critical role in the decision-making process for financial institutions, investors, and businesses. The accuracy and reliability of the analysis depend on the quality of the data and the effectiveness of the algorithms used. However, financial data poses several challenges that require specialized techniques for effective analysis. One of the major challenges in financial data analysis is the problem of imbalanced data, where the number of instances in the minority class is much lower than that of the majority class. In a number of scenarios, such as fraudulent transactions and credit/loan approval, the significant variation in distribution inconsistency among categories, ranging from tens to hundreds of times, results in poor classification performance and biased predictions. Financial institutions face significant financial losses and legal consequences if they misclassify transactions due to imbalanced data. To mitigate these risks, ML algorithms must prioritize addressing imbalanced data and ensuring unbiased decision-making.

Research on financial imbalanced data holds great significance in the field of financial data analysis. It is one of the most extensively studied areas concerning imbalanced data. The majority of studies in this domain have primarily focused on financial fraud detection, as well as property refinance prediction, loan default prediction, corporate bankruptcy prediction, and credit card approval [30]. In recent years, numerous algorithms have been proposed to address the problem of imbalanced data in this domain [18, 38, 57, 95].

The bagging method has been shown to improve the classification performance of imbalanced data for financial analysis [23, 37, 103]. However, traditional bagging does not take into account class distribution and may prioritize the majority class, leading to biased predictions. Cost-sensitive learning and weighted bagging are two approaches that have been proposed to address the challenges of imbalanced data [23, 66, 112]. Cost-sensitive learning involves assigning different costs to misclassification errors based on the class distribution and the misclassification costs. The weighted bagging method combines multiple base classifiers that were trained on different subsets of the training data, each with its own weight.

3.1.3 Our contribution

In this study, we propose a new bagging algorithm based on DC (BaggingDCA) that provides an efficient training method for various kinds of base learners and loss functions. The performance of this proposed method is carefully evaluated through extensive experimentation, comparing its effectiveness with existing methods using widely accepted benchmark datasets.

Moreover, for tackling the imbalanced data in financial problems, we introduce a new approach, named cost-sensitive weighted bagging DCA-based (CSB-DCA) combining the strengths of cost-sensitive learning and weighted bagging to improve the classification performance of

imbalanced financial data. The cost-sensitive method assigns weights to each instance of data based on its cost and importance, taking into account the class distribution and the misclassification costs, while the weighted bagging method adjusts model contributions in the final prediction to enhance the robustness and accuracy of the prediction model. We expect that this combination ultimately results in a substantial enhancement to the classification model's overall performance.

3.2 Weighted bagging approach

3.2.1 Ensemble learning and Bagging method

Bagging, introduced by L. Breiman in 1996 [13] is a ML ensemble meta-algorithm intended to increase the robustness and accuracy of ML algorithms. As an ensemble learning method, bagging can improve the performance of a given learning algorithm by combining the outputs of many classifiers via majority voting or averaging. Furthermore, by combining models trained on various bootstrap resample versions of the training set, bagging decreases variance and aids in avoiding overfitting. Although it is frequently used in conjunction with decision trees techniques, it may be applied to any type of ML algorithm.

Random forest [15], is a tree-based variant of the bagging method. Each tree in the random forest algorithm only examines a bootstrapped set of samples, and it also randomly picks subsets of features (aka. feature bootstrapping or random subspace method) used in each data sample. As the result, random forest is computationally efficient than the standard bagging approach thanks to the fact that it only needs to deal with a subset of features. In addition, feature bootstrapping also helps to improve variance by decreasing correlation between trees and the ability to rank the importance of features [4]. Brown and Mues demonstrated that random forest worked effectively when dealing with samples with a significant class imbalance in the context of credit scoring problem [16].

Bagging approaches provide a variety of advantages, such as the ability to enhance accuracy and avoid overfitting, scalability, the capability to execute parallel processing. However, the standard approach of bagging assumes that all models in the ensemble have equal prediction ability, which may not be true in practice. Equal weights strategies may not be as effective as weighted strategies for a variety of reasons, such as models may be better at predicting certain subsets of data or models in the ensemble may have different levels of accuracy.

3.2.2 Weighted bagging

Suppose there is a bootstrap aggregating model from n individual base learner (aka. weak classifier or weak leaner), the overall output of the bagging models is a weighted combination

of the individual classifier outputs. This can be represented by the following equation

$$f(x) = \sum_{j=1}^{n} w_j f_j(x) \tag{3.1}$$

where $f(x)$ is the aggregated model, $f_j(x)$ is the j^{th} classifier predictor, w_j is the weight for combining the j^{th} classifier, and x is an input vector of the ensemble model.

The weights must be determined properly in order for the model to perform effectively. Naturally, a straightforward way is to assign an equal weight to each classifier in a standard bagging scheme. Another approach is to assign each classifier a weight value according to its contribution to the aggregating model. There have been studies on weighted bagging in the literature. Breiman [13] proposed the bagging method by using a majority vote to combine multiple classifier outputs with equal weights. Wang et al. [97] proposed a dynamic weighting approach that assigns a weight to each weak learner based on their performance. Leblanc and Tibshirani [46] suggested using weights for the bagging algorithm with a non-negativity constraint.

Let y be a vector of the expected classifier outputs and $\hat{y}_j$ be the prediction from the j^{th} classifier predictor. Predictions from a set of n predictors can be put in a matrix as: $\hat{y} = [\hat{y_1}\hat{y_2}\hat{y_3}...\hat{y_n}]$ The vector of predictions from the ensemble classifier $f(X)$ can be represented as

$$y = \hat{y}w = w_1\hat{y}_1 + w_2\hat{y}_2 + \cdots + w_{\mathrm{n}}\hat{y}_{\mathrm{n}}$$

where each column corresponds to an individual predictor.

Given a training dataset $\{(x_i, y_i)\}_{i=1}^{m}$ where m is the number of samples of the training dataset, y_i is the corresponding label (either 1 or -1). We denote the output of the aggregated model at data point i^{th} as $\hat{y}_i$. In order to find an optimal set of weights, we will minimize the following optimization problem

$$\begin{aligned} &\min_{w \in R^n} K(w) = \frac{1}{m}\sum_{i=1}^{m} \ell(y_i, \hat{y}_i) = \frac{1}{m}\sum_{i=1}^{m} \ell\Big(y_i, \sum_{j=1}^{n} w_j f_j(x_i)\Big) \\ &\text{s.t. } \sum_{j=1}^{n} w_j = 1, \\ &\quad w_j \geq 0, \quad \forall j = 1, \ldots, n. \end{aligned} \tag{3.2}$$

where ℓ which is given arbitrarily by the user depending on the training purposes is a loss to measure the discrepancy between the predicted values and the true targets. The idea of finding a good combination of base learners via optimizing this discrepancy is traced back to [69] where the quadratic loss was studied. Here we aim to investigate a more general class of losses (possibly nonconvex) that can be used flexibly in many cases.

3.3 Optimization problem based on DCA

According to Zhou and Zhou [111], a loss ℓ is called an LS-DC loss if there exists a constant $A > 0$ such that the associated function ψ, i.e. $\ell(y, \hat{y}) = \psi(1 - y\hat{y})$, has the following DC decomposition

$$\psi(u) = Au^2 - \big(Au^2 - \psi(u)\big). \tag{3.3}$$

It is pointed out in [111] that almost all commonly used losses in ML are LS-DC or can be approximated by an LS-DC loss.

Let $\ell(y, t)$ be any LS - DC loss associated with ψ, the optimization problem (3.2) can be written as the following DC program

$$\begin{aligned} \min_{\boldsymbol{w}\in\mathbb{R}^n} K(w) &= \frac{1}{m}\sum_{i=1}^{m} \psi\Big(1 - y_i \sum_{j=1}^{n} w_j f_j(x_i)\Big) := G(w) - H(w) \\ \text{s.t. } & \sum_{j=1}^{n} w_j = 1, \\ & w_j \geq 0, \quad \forall j = 1, \ldots, n. \end{aligned} \tag{3.4}$$

with the DC components being given by

$$G(w) := \frac{1}{m} A \sum_{i=1}^{m} \Big(1 - y_i \sum_{j=1}^{n} w_j f_j(x_i)\Big)^2, \tag{3.5}$$

$$H(w) := \frac{1}{m} \sum_{i=1}^{m} \left(A\Big(1 - y_i \sum_{j=1}^{n} w_j f_j(x_i)\Big)^2 - \psi\Big(1 - y_i \sum_{j=1}^{n} w_j f_j(x_i)\Big) \right). \tag{3.6}$$

At iteration k, we have the subgradient of H computed as

$$\begin{aligned} \nabla H(w^k) &= \frac{1}{m}\sum_{i=1}^{m} 2A(1 - y_i \sum_{j=1}^{n} w_j^k f_j(x_i))(-y_i f_j(x_i))_{j=1,2,\ldots,n} \\ &- \frac{1}{m}\sum_{i=1}^{m} \psi'\Big(1 - y_i \sum_{j=1}^{n} w_j^k f_j(x_i)\Big)(-y_i f_j(x_i))_{j=1,2,\ldots,n} \\ &= \frac{1}{m}\sum_{i=1}^{m} \left(\psi'\big(1 - y_i \sum_{j=1}^{n} w_j^k f_j(x_i)\big) - 2A\Big(1 - y_i \sum_{j=1}^{n} w_j^k f_j(x_i)\Big) \right) (y_i f_j(x_i))_{j=1,\ldots,n}. \end{aligned} \tag{3.7}$$

where $\psi'(u)$ denotes a subgradient of ψ at u. Thus, we solve the following convex subproblem

$$\begin{aligned}
&\min_{w\in R^n} G(w) - \langle \nabla H(w^k), w\rangle = \min_{w\in R^n} \frac{A}{m}\sum_{i=1}^{m}\Big(1 - y_i w^T F(x_i)\Big)^2 - \langle \nabla H(w^k), w\rangle \\
&= \min_{w\in R^n} -\frac{2A}{m}\sum_{i=1}^{m} w^T y_i F(x_i) + \frac{A}{m}\sum_{i=1}^{m}\Big(w^T F(x_i)\Big)^2 - \langle \nabla H(w^k), w\rangle \\
&= \min_{w\in R^n} \frac{A}{m} w^T\Big(\sum_{i=1}^{m} F(x_i)F(x_i)^T\Big)w - w^T\Big(\frac{2A}{m}\sum_{i=1}^{m} y_i F(x_i) + \nabla H(w^k)\Big) \qquad (3.8)\\
&\text{s.t. } \sum_{j=1}^{n} w_j = 1, w_j \geq 0, \quad \forall j = 1,\ldots,n.
\end{aligned}$$

The optimization problem (3.8) is a quadratic program. Because w has a relatively small dimension (e.g. commonly, number of base classifiers $n < 1000$), the solution can be efficiently solved using existing packages.

3.4 BaggingDCA performance on popular benchmark dataset

3.4.1 Weighted bagging DCA algorithm

The strategy to solve (3.8) at each iteration and seek for the optimal weights of the problem 3.2 is a standard DC algorithm and is presented in Algorithm 1:

Algorithm 1 Weights calculation based on DCA

Input: Given a training set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$ with $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$;
Any LS-DC loss function $\psi(u)$ with parameter $A > 0$;
Set of n weak classifiers $F = \{f_1(x), f_2(x), \ldots, f_n(x)\}$;
Set $k = 0$, $w^0 = \mathbf{1} \in R^n$;
repeat
1. Compute $\nabla H(w^k)$ in (3.7).
2. Solve (3.8) to obtain w^{k+1}.
3. Set $k \leftarrow k + 1$.

until Stopping criterion.
Output: w

The process of the bagging method is as follows: Each base learner, using a specific technique such as decision trees, knn, or neural networks, is trained using a sample of instances taken from the original dataset by sampling with replacement. Each sample normally includes the same amount of instances as the original dataset in order to ensure a sufficient number of instances per base learner. Bootstrapping features can also be used when combining multiple base learner models, such as in bagging methods like random forests, for further reducing the variance of models and improve the overall performance. After that, a base learner is trained

on each random subset sample sequentially or more quickly through parallel training. Finally, to determine the prediction of an unseen instance, majority vote of the base learners' predictions is performed. Thus, by integrating LS-DC to the bagging scheme, we have a new bagging algorithm referred to as BaggingDCA, and is presented in the Algorithm 2:

Algorithm 2 BaggingDCA

Input: Given a training set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{m}$ with $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$;
Base learning algorithm L;
Number of base classifiers n;
for $j = 1 \ldots n$ **do**
1. Generate a bootstrap sample $\mathcal{D}_j$ from $\mathcal{D}$ by sampling with replacement (with feature bootstrapping).
2. Train a base classifier $f_j(x) = L(\mathcal{D}_j)$.
end for
3. Calculate $w \in R^n$ using algorithm 1.
4. Generate bagging model f(x)=$\sum_{j=1}^{n} w_j f_j(x)$.
Output: $f(x)$

3.4.2 BaggingDCA performance on popular benchmark dataset

In this section, we analyze the performance of our proposed algorithm BaggingDCA with two algorithms: Standard Bagging (StdBagging) and dynamic weighting bagging [97] (Authors used Lasso-logistic regression as the base learner and referred to it as LLRE; for the convenience of our experimental purposes with several base learners, we refer to the method as DWBagging). In this study, we examine three comparison algorithms with four different base learners: Decision-trees, K-Nearest neighbor, LinearSVM and Neural network. Each algorithm will perform with feature bootstrapping, that means each base model will be trained on a subset of the features. In the case of training tree-based estimators with subset of features, our bagging algorithm is the standard random forest algorithm of L. Breiman [15]. We utilize several datasets with different properties and characteristics in our experiment.

The authors of the study [111] showcased that the Least-squared loss function yields superior accuracy performance, whereas the Ramp loss proves effective in managing noisy data. Thus, in our experimental setup, we employ two types of loss functions: a convex LS-DC loss function and a nonconvex LS-DC loss function and their corresponding subdiferentials:

1. Least squares: $\psi(u) = u^2$ is a convex loss function; $\nabla\psi(u) = 2u$.

2. Smoothed ramp loss (As ramp loss is not LS-DC, this is a smoothed approximation): $\psi(u) = \begin{cases} \frac{2}{a}u_+^2, & u \leq \frac{a}{2}, \\ a - \frac{2}{a}(a-u)_+^2, & u > \frac{a}{2}, \end{cases}$ with $a > 0$ is a nonconvex function;

$$\partial\psi_a(u) = \begin{cases} \frac{4}{a}u_+, & u \leq \frac{a}{2} \\ \frac{4}{a}(a-u)_+, & u > \frac{a}{2} \end{cases}.$$

Numerical experiments will be performed with the two above loss functions: Least squares and smoothed ramp loss. Thus, we denote our algorithms as BaggingDCA1, BaggingDCA2 corresponding with BaggingDCA with least squares loss, and BaggingDCA with smoothed ramp loss.

3.4.3 Experimental setting and Datasets

The three comparison algorithms: BaggingDCA, StdBagging and DWBagging are implemented in MATLAB and executed on a PC with an Intel Core i7-8700 CPU @3.20GHz×6 16GB. The PC runs Windows 10 with MATLAB-2021a. For creating bootstrap samples with replacement to train each base learner, we use the function: *ransample(n_learner,n_learner,true)* in Matlab. For choosing feature subsets, we use the function: *ransample(m_feature,*$sqrt(m)$*,false)* with the number of subset features to draw from m features is set as $sqrt(m)$. For DCA algorithm, the constant A is set as 1, the truncated parameter a of the non-convex loss is set as 2 and the initial weights are set as a vector of all ones. The stop condition of DCA is set as 10^{-5}. To solve the quadratic problem in (3.8), we use the function **quadprog**[11] in MATLAB. The number of base-classifiers: $n = 500$ is set for all algorithms. The base learners: Decision-trees, K-Nearest neighbor, LinearSVM and Neural network for the comparison algorithms can be obtained from corresponding functions: **fitctree** [12], **fitcknn**[13], **fitclinear**[14], **fitcnet**[15] in Matlab. All parameters of the four functions are set as default values.

Datasets

Benchmark datasets from LibSVM: To assess the generalizability and scalability of the proposed algorithm, in the first experiment, we use seven well-known benchmark datasets from the LIBSVM website[16]. All datasets consist of binary data and contain a variety of features and samples. Table 3.1 summarizes the information included in the datasets. This table illustrates that the selected datasets include datasets with varying levels of size, features, and class proportion. Additional information on these datasets is available on the LIBSVM website.

[11] https://www.mathworks.com/help/optim/ug/quadprog.html
[12] https://www.mathworks.com/help/stats/fitctree.html
[13] https://fr.mathworks.com/help/stats/fitcknn.html
[14] https://fr.mathworks.com/help/stats/fitclinear.html
[15] https://fr.mathworks.com/help/stats/fitcnet.html
[16] https://www.csie.ntu.edu.tw/∼cjlin/libsvm/

Dataset	#of features	#of instances	#of label 1	#of label -1
Splice	60	3,175	1,648	1,527
Gisette	5,000	7,000	3,500	3,500
Mushroom	112	8,124	3,916	4,208
Australian	14	690	307	383
Phishing	49	10,000	5,000	5,000
Adult	123	48,840	11,687	37,153
W8a	300	49,761	1,491	48,270
Madelon	500	2,600	1,300	1,300
Mnist-1-7	784	15,170	7,877	7,293

Table 3.1: Nine benchmark datasets from LibSVM for evaluating the performance of BaggingDCA

In the experiments, for evaluation purposes, the binary datasets that have labels in $\{+1, -1\}$ will be split into two separate sets: Trainset and testset. To avoid overfitting and preserve the proportions of the classes, we use the function *StratifiedShuffleSplit* in Sklearn 1.1.1 [17] to split the dataset with proportions: 80% and 20% for trainset and testset, respectively.

3.4.4 Experiment 1: Evaluating BaggingDCA method on popular benchmark datasets

Due to the random nature of bagging algorithms, to prevent biased results caused by the random sampling of subsets of data and subsets of features, we execute each base classifier of each comparison method ten times, and then take the mean and standard deviation of the results for reporting. For comparison purpose, the accuracy on testsets of related algorithms are reported. The values that are highlighted in bold are the ones with the best results. The result of the experiment is presented in Table 3.2.

[17] https://scikit-learn.org/.../sklearn.model_selection.StratifiedShuffleSplit.html

Dataset	Algorithm	Base learner			
		Decision tree	K-NN	Linear SVM	Neural net
Splice	StdBagging	95.47±0.56	93.93±0.71	80.95±0.79	93.83±0.74
	DWBagging	95.62±0.56	94.04±0.68	81.12±0.86	94.02±0.74
	BaggingDCA1	**96.07±0.56**	94.99±0.81	85.65±1.03	94.75±0.93
	BaggingDCA2	95.90±0.52	94.99±0.81	84.99±1.05	94.72±0.95
Gisette	StdBagging	95.86±0.16	93.39±0.31	93.31±0.24	91.26±0.14
	DWBagging	95.85±0.18	93.44±0.28	93.35±0.20	91.25±0.18
	BaggingDCA1	**96.10±0.12**	93.81±0.46	94.68±0.45	92.69±0.41
	BaggingDCA2	96.07±0.25	93.81±0.46	94.26±0.44	92.43±0.50
Mushroom	StdBagging	95.85±0.75	99.11±0.37	92.55±0.89	95.94±0.78
	DWBagging	95.91±0.74	99.15±0.38	92.60±0.91	96.02±0.79
	BaggingDCA1	**99.92±0.24**	99.71±0.32	99.33±0.41	99.86±0.22
	BaggingDCA2	**99.92±0.24**	99.63±0.37	99.21±0.41	99.85±0.27
Australian	StdBagging	87.74±0.83	79.27±0.62	83.80±0.90	83.43±0.98
	DWBagging	87.74±0.83	79.34±0.77	84.01±0.80	83.43±0.60
	BaggingDCA1	**88.33±0.61**	84.23±1.24	85.77±0.85	86.86±1.33
	BaggingDCA2	88.18±0.72	84.23±1.24	84.89±0.49	86.79±1.47
Phishing	StdBagging	98.91±0.12	99.36±0.15	96.60±0.63	97.97±0.45
	DWBagging	99.01±0.10	99.40±0.16	96.85±0.66	98.12±0.42
	BaggingDCA1	**100.00±0.00**	99.99±0.03	99.98±0.05	**100.00±0.00**
	BaggingDCA2	**100.00±0.00**	99.99±0.03	99.98±0.06	**100.00±0.00**
Adult	StdBagging	77.09±0.05	76.20±0.30	76.07±0.00	76.07±0.00
	DWBagging	77.23±0.10	76.19±0.27	76.07±0.00	76.27±0.06
	BaggingDCA1	**83.87±0.22**	83.63±0.23	83.44±0.24	83.30±0.37
	BaggingDCA2	83.76±0.22	82.91±0.41	82.96±0.35	83.14±0.38
W8a	StdBagging	97.06±0.01	97.07±0.03	97.04±0.02	97.06±0.01
	DWBagging	97.16±0.03	97.07±0.03	97.04±0.02	97.10±0.04
	BaggingDCA1	**98.07±0.13**	97.44±0.09	97.46±0.13	97.49±0.17
	BaggingDCA2	97.82±0.09	97.65±0.09	97.65±0.06	97.86±0.12
Madelon	StdBagging	65.19±1.41	63.04±2.01	61.00±0.85	59.38±0.86
	DWBagging	65.63±1.67	62.98±1.86	60.98±0.74	59.78±1.24
	BaggingDCA1	**71.25±1.34**	70.27±1.91	61.62±0.71	61.42±0.57
	BaggingDCA2	71.22±1.24	70.27±1.91	61.27±0.62	61.31±0.81
Mnist-1-7	StdBagging	99.60±0.05	99.51±0.01	98.95±0.01	99.55±0.04
	DWBagging	99.61±0.03	99.51±0.01	98.94±0.03	99.55±0.04
	BaggingDCA1	**99.74±0.05**	99.70±0.04	99.43±0.09	99.74±0.10
	BaggingDCA2	**99.74±0.05**	99.70±0.04	99.39±0.06	99.74±0.10

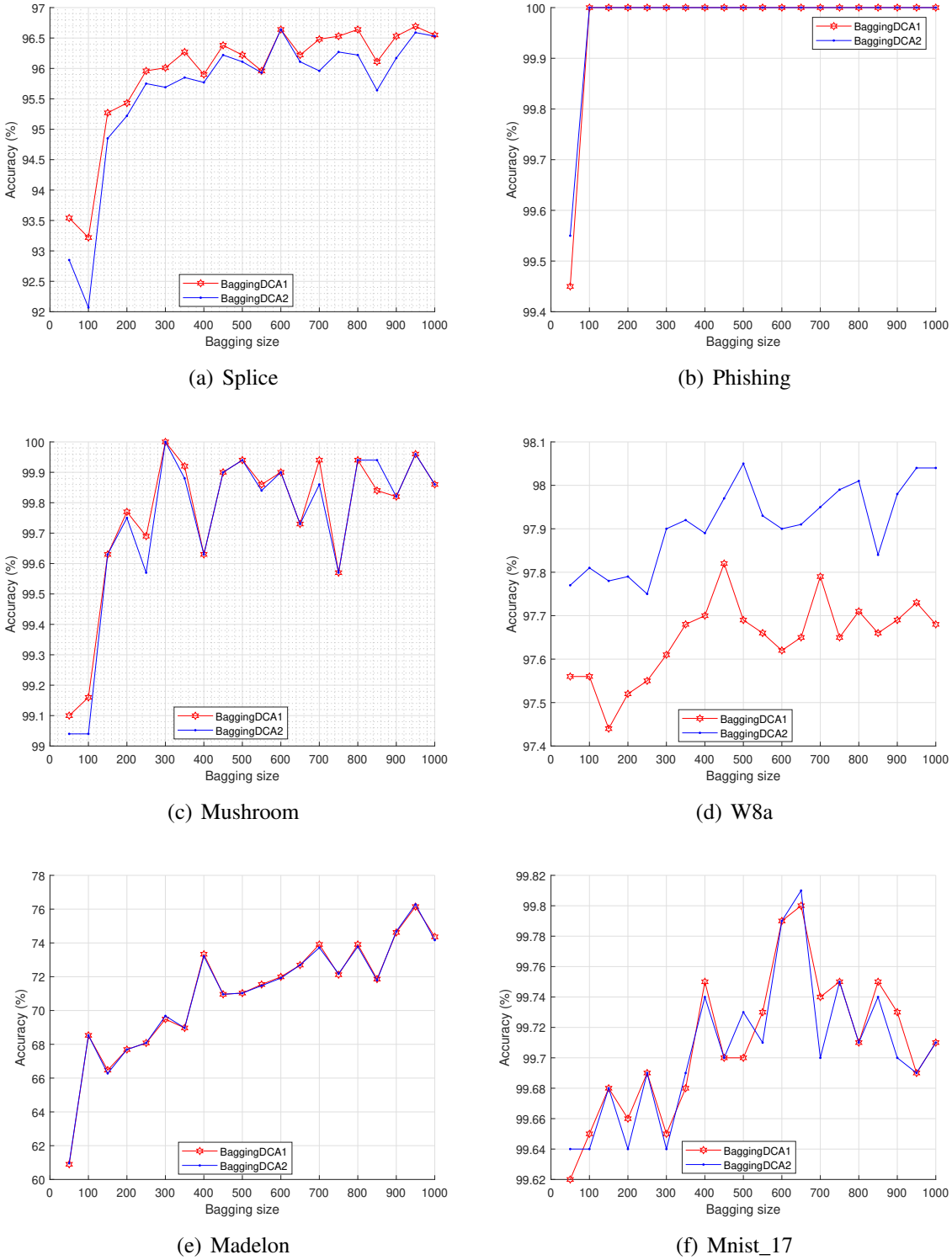
Table 3.2: Performance comparison (mean accuracy±std.) of related algorithms on nine benchmark datasets.

Comments: Table 3.2 compares the average accuracy of all comparison methods. Based on the experimental result, it is evident that our BaggingDCA algorithms give better results than StdBagging and DWBagging in most cases on nine benchmark datasets. There are cases where

our algorithm is approximately 4% to 6% more accurate than the StdBagging and DWBagging techniques, such as in the Mushroom and Adult datasets, or Madelon in the case of Decision tree. When it comes to the base classifiers, Tree-based models typically produce superior outcomes to K-nearest neighbor, LinearSVM and Neural network. Considering the comparative results of the two loss functions, the method utilizing the least squares loss function performs slightly better than the method using the smoothed ramp loss function in the majority datasets.

3.4.5 Experiment 2: Parameter analysis with different bagging sizes

In this experiment, we investigate the relationship between the bagging size and the accuracy of machine learning models, with a focus on two proposed algorithms: BaggingDCA1 and BaggingDCA2. The bagging size, or the number of models included in the ensemble, is a critical parameter in bagging strategies. The result of this experiment can help us to better understand the impact of bagging size on model accuracy. It also offers practical guidance for model hyperparameter tuning in real-world ML models. In this experiment, the base learner chosen is the decision tree algorithm. By varying the size of trees from 50 to 1,000, we intend to investigate the impact of base-learner size on the overall effectiveness of the proposed algorithms on six benchmark datasets. The result of this experiment is depicted in Figure 3.1.



(a) Splice (b) Phishing

(c) Mushroom (d) W8a

(e) Madelon (f) Mnist_17

Figure 3.1: Performances of related algorithms on benchmark datasets as the number of base-learners increases.

Comments: The following conclusions can be drawn from the experimental results depicted in Figure 3.1:

As the bagging size increases, the general tendency for both models BaggingDCA1 and

BaggingDCA2, is to experience an enhancement in accuracy. However, the rate and intensity of the accuracy increase are dataset-dependent. In other words, each dataset responds differently to an increase in bagging size, exhibiting varying levels of improvement in model accuracy. This might be due to the peculiar characteristics and inherent complexity of each dataset.

Interestingly, a further increase in bagging size beyond a certain threshold does not always lead to an increase in accuracy. In some cases, when increasing the bagging size to a certain size, the accuracy may plateau or even decrease slightly (e.g. in Mushroom, W8a, or Mnist_17). This phenomenon aligns with the general principles of machine learning and can be attributed to the model's inability to learn any new patterns from the added bags due to their redundant or noisy information.

3.5 Cost-sensitive Weighted Bagging DCA-based algorithm for imbalanced data

3.5.1 Cost-sensitive learning

The study [25] introduced a cost-sensitive method that incorporates a cost matrix to account for the higher cost associated with misclassifying instances from the minority class compared to the majority class. Denoting the loss function of a base classifier as L, the authors introduce the cost matrix C that assigns cost entry (i,j) to predict class i when the true class is j, where $i = j$ indicates a correct prediction and $i \neq j$ denotes an incorrect prediction. The optimal prediction for an example x minimizes the loss function:

$$L(x, i) = \sum_{j} P(j \mid x)C(i, j)$$

For each value of i, $L(x, i)$ represents a sum over the alternative possibilities for the true class of x. In this framework, the learning algorithm aims to produce a classifier that can estimate the probability $P(j \mid x)$ for each class j to be the true class of x. When making a prediction i for example x, the algorithm assumes that i is the true class of x.

3.5.2 Cost-sensitive weighted bagging algorithm

By incorporating the DC algorithm into the cost-sensitive bagging framework, a novel bagging algorithm called CSB-DCA is introduced. The detailed steps of CSB-DCA are presented in Algorithm 3:

Algorithm 3 CSB-DCA

Input: Given a training set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$ with $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$;
Base learning algorithm B;
Number of base classifiers n;
Any LS-DC loss function $\psi(u)$ with parameter $A > 0$;
for $j = 1 \ldots n$ **do**
1. Generate a bootstrap sample $\mathcal{D}_j$ from $\mathcal{D}$ by sampling with replacement(with/without bootstrap features).
2. Calculate cost matrix $C_j = \begin{pmatrix} C_j^{TN} & C_j^{FN} \\ C_j^{NP} & C_j^{TP} \end{pmatrix}$ with respect to $\mathcal{D}_j$.
3. Train a cost-sensitive base classifier $f_j(x) = B(\mathcal{D}_j)$ with cost C_j.
end for
4. Set $k = 0$, $w = \mathbf{1} \in R^n$; $F = \{f_1(x), f_2(x), \ldots, f_n(x)\}$;
repeat
5. Compute $\nabla H(w^k)$ in (3.7).
6. Solve (3.8) to obtain w^{k+1}.
7. Set $k \leftarrow k + 1$.
until Stopping criterion.
7. Generate bagging model f(x)=$\sum_{j=1}^n w_j f_j(x)$.
Output: $f(x)$

3.5.3 Experimental setting

To evaluate the effectiveness of the proposed CSB-DCA algorithm, we utilize several financial datasets, including credit approval, fraud detection, and bankruptcy prediction. We compare the performance of the proposed algorithm with two existing approaches, including traditional bagging and cost-sensitive bagging. The cost-sensitive bagging is the traditional bagging with cost-sensitive learning [25].

In Experiment 1, we have demonstrated that tree-based bagging outperforms other weak classifiers, including Decision Trees, K-Nearest Neighbors, Logistic Regression, Support Vector Machines, and Neural Networks. In this experiment, we adopt the decision tree as the base classifier for all comparison algorithms. When training a base classifier, the data is sampled with replacement, while the features are sampled without replacement or bootstrap features. Additionally, in the case of a tree-based classifier with bootstrap features, this approach corresponds to the well-known random forest algorithm introduced by L. Breiman [15]. All comparison algorithms were implemented using MATLAB and executed on a PC equipped with an Intel Core i7-8700 CPU @ 3.20GHz (6 cores) and 16GB of RAM. The PC ran Windows 10 operating system with MATLAB-2021a. All setups related to DCA's settings are the same in [71]. To set-up the misclassification cost matrix in the decision tree algorithm, we use the built-in parameter "$Cost$" in the fitctree function. For simple, in this study, we use the misclassification matrix $C = \begin{pmatrix} 0 & 1 \\ imb.ratio & 0 \end{pmatrix}$, where $imb.ratio = \frac{N_{\text{majority instances}}}{N_{\text{minority instances}}}$.

LS-DC losses

Similarly to Experiment 1, in this experiment, the CSB-DCA algorithm employs two loss functions: Least squares loss and Smoothed ramp loss. Consequently, we label our algorithms as CSB-DCA1 and CSB-DCA2, signifying Cost-sensitive weighted bagging DCA with Least squares loss and Cost-sensitive weighted bagging DCA with Smoothed ramp loss, respectively.

Performance measures

Evaluating the performance of imbalanced data requires careful consideration due to the unequal distribution of classes. To obtain a comprehensive understanding of the model's effectiveness, evaluating performance on unbalanced data may necessitate a combination of distinct methods. In this study, we employ four metrics: Accuracy, F-score, G-mean and Area Under the ROC Curve-AUC to evaluate the performance of our algorithm and comparison algorithms.

1. Accuracy: Traditionally, accuracy can be a simple metric for analyzing the model's correctness. The accuracy of classification is determined by:
$\text{Accuracy } = \frac{TP+TN}{TP+FP+FN+TN}$

2. F-Score:

Precision: Refers to the ratio between true positives (TP) and overall positive predictions. Simply, it informs us of the rate of the positive prediction is actually positive. $\text{Precision } = \frac{TP}{TP+FP}$

Recall: Recall (also known as Sensitivity) indicates the proportion of positive samples that are predicted to be positive. $\text{Recall } = \frac{TP}{TP+FN}$

F-Score: F-Score (also known as the F1 score) provides a balanced assessment of the model's performance, taking into account both precision and recall (the harmonic mean). $\text{F-Score } = 2 * \frac{Precision*Recall}{Precision+Recall}$

3. G-mean: G-mean [26] represents the geometric mean of the true rates, which is defined as $G - \text{ mean } = \sqrt{\frac{TP}{TP+FN} \times \frac{TP}{TP+FP}} = \sqrt{Recall * Precision}$

4. AUC: The AUC is a measure of the model's ability to distinguish between positive and negative classes. The ROC (Receiver Operating Characteristics) curve is a graphical representation of the model's performance, plotting the true positive rate versus the false positive rate for various threshold settings. To calculate AUC in Matlab, we use the trapezoidal rule in the function trapz [18].

[18] https://www.mathworks.com/help/matlab/ref/trapz.html

3.5.4 Imbalanced financial datasets

The Luxembourg banking client database

To evaluate the effectiveness of our algorithm, we have access to a real-world customer database, the Luxembourg banking client database, obtained from a commercial bank located in Luxembourg. Our original database contains 72 variables that are readily available within the bank's information system. The preliminary work required file merges, information reconciliations, meetings to understand data quality and computer coding. Then, we can characterize the customer information as follows:
- Information K.Y.C (Know you customer) on the economic, legal, tax criteria.
- Financial risk information (Probability of default – accounting information...).
- Management control information or controlling (Customer profitability history and marketing segment).
- Information on the operation of current and joint accounts (Savings – number of loans).
- Technical information related to the decision to accept or refuse the overdraft on the current account.

Datasets from UCI and Kaggle

Real-world financial datasets are often limited in availability due to the sensitive nature of financial data. In this research, we augment the evaluation of the CSB-DCA method by incorporating two datasets from UC Irvine Machine Learning Repository [19] and two datasets from Kaggle[2021], which were publicly accessible and extensively utilized by the ML community for validation purposes.

UCI Polish companies bankruptcy: This dataset's purpose is to forecast the bankruptcy of Polish companies. The data was obtained from the Emerging Markets Information Service (EMIS), a comprehensive database that provides information on emerging markets worldwide.

UCI default of credit card clients: This dataset aims to investigate the payment default patterns of Taiwanese consumers and evaluate the predictive accuracy of the probability of default using six distinct data mining techniques.

Kaggle credit card approval: This dataset utilizes credit scorecards in the financial industry to predict future default probabilities and credit card usage using personal and applicant data. It allows banks to make informed decisions (good or bad clients) regarding the credit card issuance of applicants. The original dataset consists of 18 columns, and we followed the data cleaning process described in 20 to prepare the dataset for evaluation.

Kaggle credit card fraud detection: Contains credit card transactions made by European cardholders in September 2013, consisting of 284,807 transactions over a two-day period, with

[19] https://archive.ics.uci.edu/ml/index.php
[20] https://www.kaggle.com/datasets/rikdifos/credit-card-approval-prediction
[21] https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

492 identified as frauds, indicating a highly unbalanced dataset where frauds make up only 0.172% of the total transactions. The "Time" column is excluded as a feature in our evaluation model for this dataset.

Table 3.3 summarizes the information included in all the datasets, including the Luxembourg credit approval dataset.

Table 3.3: Datasets for evaluation (dataset, abbreviation(abbr.), feature(fea.), instance(ins.), positive(pos.), negative(neg.), class imbalance ratio(imb.ratio)).

dataset	abbr.	#fea.	#ins.	#pos.	#neg.	imb.ratio
Luxembourg credit dataset	LuxCre	24	29,162	1,501	27,661	18.43
UCI Polish companies bankruptcy	UCI-1	64	7,027	271	6,756	24.93
UCI default of credit card clients	UCI-2	15	29,901	6,621	23.280	3.51
Kaggle credit card approval	Kaggle-1	43	25,134	422	24,712	58.56
Kaggle credit card fraud detection	Kaggle-2	29	284,807	492	284,315	577.88

3.5.5 Experiment 3: Data exploration and feature selection with the statistical test on Luxembourg Credit approval dataset

To understand the decisions of the credit approval process, we performed two statistics analyses: F-test and T-tests (ANOVA; Means) to dissociate variables between decisions. While the T-test helps to explain how significant the differences between group means are, the main idea behind ANOVA for feature selection is to test the statistical significance of each feature's contribution to the response feature. The objective of these tests is to identify the relevant variables to be considered in the decision-making process and to exclude features that are not significantly related to the response feature, hence lowering the model's complexity and maybe enhancing its performance. The data analysis results are presented in Figure 3.2.

Comment: After examining the variables, we determined in Figure 3.2 that the following factors are more likely to result in account approval:
- Low probability of bankruptcy (probabilities are calculated using logistic regression methods).
- Client has assets: Other accounts and investments.
- Active client: An active client is a profitable client in the sense of management control.
- Customer has many commitments with the bank (MT REVENU TOTAL, NB DOSCRED): It is more difficult to decline when the company has to repay its debts.
- Client seniority matters: It is easier to say yes to a long-time client.
- When the account operating indicators are good (average account balance – number of days of excess...).

Moreover, the χ^2 test result shows that there is no seasonality (alpha below 5%).

MAY2016 ME 2	PROBABILITE_DEFAUT	GLOBAL_ASSETS	IND_ACTIF	MT_REVENU_TOTAL	Nb_Doscred	Anciencpt	V11_UNI	V32_UNI
TESTS STATS								
Average (moment 1)	348.459854	551928.673	0.890510949	362.2854526	3.845394737	4785.841849	12.64356436	45984.36697
STD (moment 2)	459.7399214	4084683.761	0.312632382	633.0652879	2.611053903	2965.387072	20.02390818	418224.4541
SKEWNESS (moment 3)	4.783228473	8.903398887	-2.510427965	3.081278205	1.066063794	0.599214877	4.227925726	8.366168598
KURTOSIS (moment 4)	34.31135983	77.82435474	4.323262855	14.81759534	0.518149866	-0.688532995	23.1251438	71.56681078
MIN	0	0	0	-838.01	1	137	1	-423555
MAX	3972	37274908	1	5747.3	10	10743	165	3730762
QUARTILE 1	65	2870.22	1	8.51	2	2058	4	-1909.77
QUARTILE 2	197	17613.32	1	124	3	4108	7	1751.825
QUARTILE 3	562	77648.57	1	284.8	5	6530	12	15415.51
COUNT	411	411	411	411	304	411	202	411

MAY2016 ME 3	PROBABILITE_DEFAUT	GLOBAL_ASSETS	IND_ACTIF	MT_REVENU_TOTAL	Nb_Doscred	Anciencpt	V11_UNI	V32_UNI
TESTS STATS								
Average (moment 1)	3771.25	14614.58375	1	-122.81875	2	2900.0625	189.25	6848.411411
STD (moment 2)	2948.44091	51372.0546	0	1088.803461	0.40824829	920.4039308	322.3668506	25799.30278
SKEWNESS (moment 3)	0.869977948	3.997884201	#DIV/0!	-3.809412066	0	1.992101674	3.381454729	3.993858289
KURTOSIS (moment 4)	1.172096849	15.98802992	#DIV/0!	14.95746819	6	4.383715879	11.58244366	15.96505832
MIN	23	25.68	1	-4149.52	1	2057	65	-407.151
MAX	9999	207230.1	1	431.54	3	5572	1206	103552.3
QUARTILE 1	3096.25	792.06	1	0	2	2244	65	-27.427215
QUARTILE 2	3972	2447.24	1	34.85	2	2880	80	99.14738
QUARTILE 3	3972	2447.24	1	255.39	2	2880	145	1407.0965
COUNT	16	16	16	16	13	16	12	16

	PROBABILITE_DEFAUT	GLOBAL_ASSETS	IND_ACTIF	MT_REVENU_TOTAL	Nb_Doscred	Anciencpt	V11_UNI	V32_UNI
F TEST DISPERSION	8.51064E-72	1.82353E-26	#DIV/0!	0.000339464	2.67752E-08	7.26329E-06	6.0722E-112	4.06912E-16
T TEST ESPERANCE	0.000318177	0.008085046	5.53935E-12	0.096428218	1.8588E-15	1.24179E-07	0.084284597	0.070960275

Figure 3.2: F-test and T-test results.

The Luxembourg credit approval dataset

After conducting a comprehensive exploratory data analysis, we carefully selected 24 of the original 72 variables as crucial features for training our prediction model. Figure 3.3 illustrates the correlation matrix, which depicts the intricate interconnections and dependencies among the selected variables, and Table 3.4 provides a comprehensive overview of the selected variables, including their descriptive statistics.

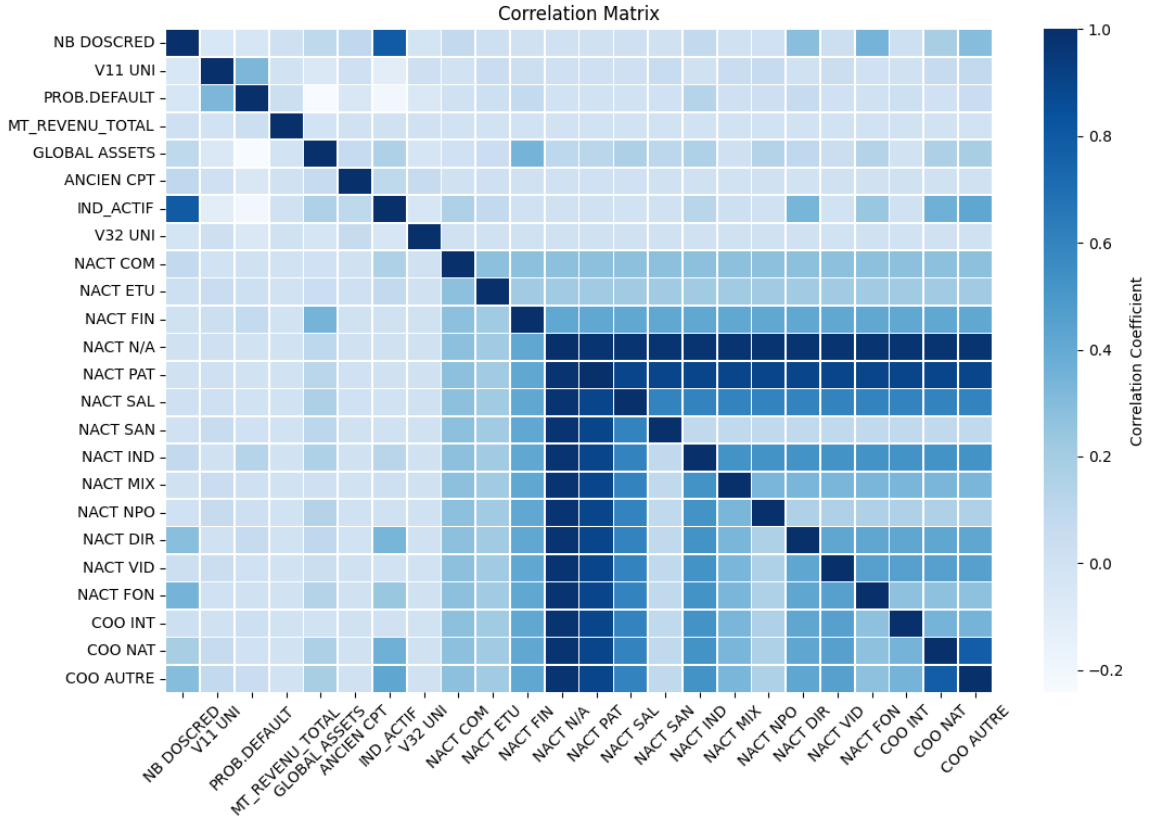


Figure 3.3: Features correlation matrix of the Luxembourg credit approval dataset. Both the X and Y axes show different attributes present in the dataset.

Variable	Observations	Missing	Minimum	Maximum	Mean	StDev.
NB DOSCRED	29,162	0	0	10	3.562	2.660
V11 UNI	29,162	0	1	1,206	38.027	67.028
PROBABILITY DE DEFAULT	29,162	0	0	9,999	647.764	1115.465
MT REVENU TOTAL	29,162	0	-4,149	3727490.8	617.5	29559.8
GLOBAL ASSETS	29,162	0	0	37274908.0	6335689.3	10537986.4
ANCIEN CPT	29,162	0	137	10,743	5474.530	2383.977
INDICATEUR ACTIF	29,162	0	0	2	1.567	0.508
V32 UNI	29,162	0	-423555.0	3730762.0	57137.8	385265.8
NACT COM	29,162	0	0	1	0.794	0.404
NACT ETU	29,162	0	0	1	0.421	0.494
NACT FIN	29,162	0	0	1	0.538	0.499
NACT N/A	29,162	0	0	1	0.561	0.496
NACT PAT	29,162	0	0	1	0.550	0.497
NACT SAL	29,162	0	0	1	0.515	0.500
NACT SAN	29,162	0	0	1	0.279	0.448
NACT IND	29,162	0	0	1	0.605	0.489
NACT MIX	29,162	0	0	1	0.355	0.479
NACT NPO	29,162	0	0	1	0.321	0.467
NACT DIR	29,162	0	0	1	0.694	0.461
NACT VID	29,162	0	0	1	0.290	0.454
NACT FON	29,162	0	0	1	0.320	0.466
COO INT	29,162	0	0	1	0.117	0.322
COO NAT	29,162	0	0	1	0.485	0.500
COO AUTRE	29,162	0	0	1	0.401	0.490

Table 3.4: Statistics descriptive of the Luxembourg credit approval dataset.

The dataset exhibits a skewed distribution, consisting of 27,661 samples labeled as approval and 1,501 samples labeled as refusal. The labels in this context represent the outcomes of the

customer credit approval process, indicating whether an overdraft is approved or not. Specifically, we assign the labels in {-1, 1} to represent the outcomes, where -1 indicates an approved overdraft and 1 indicates a disapproved overdraft.

3.5.6 Experiment 4: Assessing the efficacy of the CSB-DCA method in handling imbalanced data across varied financial datasets

In order to assess the effectiveness of the proposed algorithm, we conduct comprehensive experiments on five datasets. For evaluation, we employ 10-fold cross-validation for all the algorithms in order to assess their performance comprehensively and reduce any potential bias resulting from a single train-test divide. The data is partitioned into ten subsets or “folds", and the algorithms were trained on nine folds and tested on the remaining fold after each iteration. The test sets were sequentially rotated to ensure that all data points were both trained and tested exactly once following the scheme of n-fold cross-validation. The average results including: Accuracy, F-score, G-mean, AUC across the ten folds will be reported in our analysis. The result of this experiment is presented in Table 3.5.

Dataset	Algorithm	**Accuracy**	**Fscore**	**Gmean**	**AUC**
LuxCre	StdRF	99.66	96.62	96.66	96.67
	CSRF	99.94	99.28	99.34	99.34
	CSB-DCA1	**99.98**	**99.75**	**99.78**	**99.78**
	CSB-DCA2	**99.98**	**99.75**	**99.78**	**99.78**
UCI-1	StdRF	96.14	NaN	3.51	50.06
	CSRF	96.16	NaN	6.07	50.58
	CSB-DCA1	**97.55**	**53.98**	**61.53**	**68.91**
	CSB-DCA2	**97.57**	**54.68**	**62.13**	**69.28**
UCI-2	StdRF	78.97	28.35	41.67	57.83
	CSRF	80.19	46.94	61.33	65.75
	CSB-DCA1	**80.54**	**49.94**	**63.19**	**67.42**
	CSB-DCA2	**80.55**	**50.92**	**64.25**	**68.05**
Kaggle-1	StdRF	98.32	NaN	0.00	50.00
	CSRF	98.77	53.59	65.28	71.23
	CSB-DCA1	**100.00**	**100.00**	**100.00**	**100.00**
	CSB-DCA2	**100.00**	**100.00**	**100.00**	**100.00**
Kaggle-2	StdRF	99.93	73.66	77.48	80.01
	CSRF	99.93	75.96	80.39	82.31
	CSB-DCA1	**99.95**	**83.74**	**86.64**	**87.53**
	CSB-DCA2	**99.95**	**83.97**	**87.07**	**87.90**

Table 3.5: Performance of the four algorithms in handling imbalanced data across five financial datasets

Comments: The following conclusions can be drawn from the experimental results depicted in Table 3.5:

The experimental results on the LuxCre dataset indicate that the cost-sensitive algorithms, CSRF, CSB-DCA1, and CSB-DCA2, exhibit significantly higher performance in all four evaluation metrics compared to the standard algorithm, StdRF. This finding clearly proves that the performance of the bagging algorithm on imbalanced financial datasets has been greatly improved when a cost-sensitive technique is included.

Moreover, the CSB-DCA1 and CSB-DCA2 algorithms consistently outperform the StdRF and CSRF algorithms across all evaluation metrics in all five datasets. They achieve higher accuracy, F-score, G-mean, and AUC scores, indicating their improved and superior performance in classifying the respective datasets. Notably, the CSB-DCA1 and CSB-DCA2 algorithms demonstrate flawless performance in classifying the UCI-1 and Kaggle-1 datasets, while the StdRF and CSRF algorithms give low performance in F-score, G-mean and AUC. Specifically, for the Kaggle-1 dataset, CSB-DCA1 and CSB-DCA2 algorithms demonstrate superior performance (100%) in all four evaluation metrics compared to StdRF and CSRF. The results clearly demonstrate that the weighted bagging algorithm based on DCA has outperformed standard bagging algorithms.

F-Score as NaN: For the algorithms StdRF and CSRF, the F-Score is reported as NaN in some cases. When calculating the F-Score, if the values of TP, FP, or FN are equal to 0, or both Precision and Recall are equal to 0, the result of the F-Score will be NaN. This occurs when the dataset has no positive predictions (True Positive: TP = 0) or when there are no accurately classified positive predictions (both Precision and Recall = 0). In the financial domain, positive instances could represent fraudulent transactions, high-risk customers, or other important events that require attention or action. If a classifier in a financial organization gives a low or zero TP, it means that the classifier is not effectively identifying positive instances correctly.

The presence of datasets with high accuracy but low F-score, G-mean, and AUC (UCI-1, Kaggle-1, Kaggle-2) demonstrates the significance of incorporating diverse evaluation metrics, such as F-score, G-mean, and AUC in addition to accuracy to provide a comprehensive and unbiased evaluation of the models.

3.5.7 Experiment 5: Evaluating the robustness of the CSB-DCA method on imbalanced datasets

The purpose of this experiment is to assess the resilience of the CSB-DCA method through the introduction of noise to pre-existing datasets. We expect to gain a deeper comprehension of the performance of the CSB-DCA method and its capacity to effectively handle noisy and imbalanced data in a wide variety of financial contexts.

Adding random features to a dataset in ML testing refers to the process of including irrelevant or meaningless features in the dataset. A robust algorithm should be able to identify and disregard these irrelevant features, while still accurately predicting the outcome variable. If the algorithm is unable to do so, it may be overfitting the data and may not generalize well to new

data. For this purpose, we introduce 5 new features to all the evaluation datasets by using the function *normrnd* in Matlab following the standard normal distribution (mean = 0, standard deviation = 1) and perform the experiment in the same way with Experiment 1.

Dataset	Algorithm	**Accuracy**	**Fscore**	**Gmean**	**AUC**
LuxCre	StdRF	99.01	89.32	89.82	90.34
	CSRF	99.67	96.67	96.71	96.77
	CSB-DCA1	**99.97**	**99.75**	**99.75**	**99.76**
	CSB-DCA2	**99.97**	**99.75**	**99.75**	**99.76**
UCI-1	StdRF	96.13	NaN	0	50
	CSRF	96.16	NaN	5.05	50.29
	CSB-DCA1	**97.48**	**53.43**	**61.81**	**68.87**
	CSB-DCA2	**97.52**	**54.31**	**62.03**	**69.72**
UCI-2	StdRF	78.86	21.35	35.17	55.59
	CSRF	80.10	42.62	55.87	63.40
	CSB-DCA1	**81.68**	**47.95**	**60.03**	**66.12**
	CSB-DCA2	**81.72**	**49.25**	**61.76**	**67.24**
Kaggle-1	StdRF	98.32	NaN	0	50
	CSRF	98.32	NaN	4.87	50.12
	CSB-DCA1	**100.00**	**100.00**	**100.00**	**100.00**
	CSB-DCA2	**100.00**	**100.00**	**100.00**	**100.00**
Kaggle-2	StdRF	99.93	74.04	78.08	80.48
	CSRF	99.93	75.91	80.77	82.62
	CSB-DCA1	**99.95**	**83.55**	**86.92**	**87.56**
	CSB-DCA2	**99.95**	**83.61**	**87.05**	**88.05**

Table 3.6: The performance of four algorithms on five financial datasets after introducing five random features.

Comments on the experimental results: As shown in Table 3.6, the proposed CSB-DCA algorithms were able to maintain stable performance metrics across the scenarios tested in Experiment 4, in contrast to the other baselines. Specifically, while the StdRF and CSRF approaches generally exhibited decreases in their evaluation metrics for most imbalanced datasets, the CSB-DCA methods demonstrated robustness by consistently achieving comparable results. (In the case of UCI-2 AND Kaggle-2, there is a slight increase in some metrics, however, the difference is not significant).

3.6 Conclusion

In this chapter, we propose two new bagging algorithms based on the DCA scheme: DCBagging, which is applicable to the majority of existing loss functions for both classification and regression problems, and CSB-DCA, a variant of BaggingDCA that is combined with Cost-sensitive techniques for solving the imbalanced financial data. The proposed algorithms inherit

the benefits of standard bagging, such as decreasing variance, the ability to set up parallel computations and helping to avoid overfitting, additionally, it improves the accuracy of the training model.

The experiments were designed to thoroughly test the efficacy and robustness of the proposed methods on various datasets, on both popular benchmark and real-world datasets across multiple financial domains, such as fraud detection, credit approval, and bankruptcy company prediction. By maintaining stable and in many cases improved performance relative to the baselines, the results validate that our bagging methods not only enhance overall prediction accuracy, but also impart greater robustness to issues commonly faced when applying machine learning to many classification problems, especially imbalanced financial problems.

However, there are some other studies that employ alternative ensemble approaches with different base classifiers, such as pasting, boosting, and stacking with various types of base learners: Logistic regression, Deep Neural network, or a mixture of learners in [14, 35, 59, 82]. In the future, we will intensively study whether our proposed algorithm may improve these types of ensemble learning problems.

Chapter 4

Deep transfer learning with MCSDCA and applications in Healthcare problems

Abstract. In this chapter, we provide a foundational understanding of Deep Learning and Deep transfer learning, establishing the groundwork for our proposed method: a hybrid CNN-BiLSTM model enhanced by the Markov Chain Stochastic DCA (MCSDCA) algorithm and pre-trained language models. We apply this approach to address specific challenges in text classification within the healthcare domain. The hybrid CNN-BiLSTM architecture enables the classification model to incorporate both local and sequential information from the text input data. Consequently, this empowers the model to grasp more intricate connections among words and attain a higher level of precision in understanding the text's meaning. The presented case studies highlight the effectiveness of the proposed approach in predicting cancer, classifying medical specialties from textual data, and classifying clinical actions based on discharge summaries. The proposed methodology introduces the transfer learning framework with three common pre-trained language models: Word2Vec, BERT, and RoBERTa. Experimental comparisons between CNN, LSTM, and the proposed method are conducted, along with assessments of existing deep-learning optimizers and pre-trained language models.

4.1 Introduction

Healthcare is characterized by its complexity, where patient data may come in diverse forms of textual data, ranging from Electronic Health Records (EHRs), and discharge summaries, and prescriptions to genetic sequences and medical reports. The task of efficiently arranging and comprehending this overwhelming amount of diverse unstructured data poses a considerable obstacle. The utilization of Natural Language Processing (NLP) methodologies is frequently employed to navigate the intricacies associated with this complexity. When combined with Machine Learning (ML), Natural Language Processing (NLP) has the capability to extract patterns, evaluate semantic meanings, and produce responses from diverse forms of unstructured data. However, the traditional approach of developing specialized models for each task often

requires substantial amounts of annotated data, which can be scarce and expensive to obtain in the medical domain. This is particularly true for tasks that involve rare diseases or conditions. Moreover, the rapid evolution of medical knowledge necessitates models that can adapt and generalize effectively across various healthcare challenges.

Transfer learning based on Deep learning (DL) (Aka. Deep transfer learning) offers a compelling solution to these challenges. By leveraging knowledge learned from related domains or tasks, transfer learning enables models to overcome data scarcity and improve performance. For example, pre-trained language models such as Word2Vec, BERT, AlBerta, and GPT are trained on enormous general domain text and learn rich semantic and syntactic understanding. These pre-trained models then act as strong baselines that can be fine-tuned for clinical NLP applications using relatively small labeled datasets. By initializing models with knowledge extracted from large source domains, transfer learning enables information to successfully transfer to different but relevant target domains that may have limited labeled data available. This effective technique enables deep learning models to generalize effectively even when training data is scarce.

Related work: In recent years, there has been a substantial amount of research conducted in the healthcare domain utilizing deep transfer learning techniques. Turner et al. [92] employed Word2Vec as text representation for text classification tasks to categorize lupus phenotypes. Ayoub et al. proposed a new end-to-end technique [31] for the construction of a biomedical knowledge graph from clinical textual using a variation of BERT models. Li et al. [50] demonstrated that BERT-based models have reached the state-of-the-art for biomedical and clinical entity normalization. Numerous studies have been conducted to fined-tune general pre-trained language models in order to develop specialized pre-trained language models in healthcare and medical disciplines, for example, BioWordVec [107], Clinical Bert[3], BioBert [49], BioGPT[54].

Pre-trained language models refer to DL models that have undergone prior training and are readily applicable for a range of language processing applications. In order to leverage the capabilities of deep transfer learning through pre-trained language models, there exist various methodologies, with feature-based and fine-tuning being the two predominant techniques. The training process of pre-trained language models, which rely on artificial neural networks, follows to the standard training approach of DL. Luu et al. [55] proposed a new stochastic algorithm called Markov chain stochastic DCA (MCSDCA) to handle a class of stochastic (nonsmooth) DC programs with endogenous uncertainty in the absence of i.i.d. samples for training deep neural networks. This approach to DL is completely novel and extremely promising. The authors have provided evidence to support the superiority of the MCSDCA model over existing techniques. However, the researchers exclusively employed classical DL models for the purpose of addressing time series prediction challenges in their study. In comparison to classical DL models, this study will focus on a comprehensive evaluation of this new optimizer for advanced DL schemes in Deep transfer learning for solving some problems in Healthcare.

More specifically, in this study, we make contributions that can be summarized as follows:

1. We propose a new hybrid CNN-LSTM network architecture designed to extract both local and sequential features from healthcare text data.

2. We conduct a thorough evaluation of the Markov chain stochastic DCA algorithm in conjunction with the proposed architecture, utilizing diverse datasets and deep learning algorithms.

3. Lastly, we address the scarcity problem in healthcare by integrating the proposed approach with state-of-the-art pre-trained language models.

4.2 Background

To lay the foundation, we introduce the fundamental concepts of Deep learning and Deep transfer learning in NLP, explaining its principles and methodologies. We discuss how transfer learning enables models trained on one task to be repurposed for related tasks, significantly improving efficiency and performance in this section.

4.2.1 Deep transfer learning in NLP

Deep learning (DL), as its name suggests, is an ML subfield that concentrates on training artificial neural networks, which are composed of multiple layers of interconnected nodes or neurons to carry out complex tasks.

The depth of DL, which is represented as L signifies the number of layers in the neural network. Mathematically, a deep neural network can be represented as:

$$f(X) = f^{(L)}(f^{(L-1)}(\dots f^{(2)}(f^{(1)}(X;\Theta^{(1)});\Theta^{(2)})\dots;\Theta^{(L-1)});\Theta^{(L)})$$

where X is the input data, $f(X)$ is the output of the deep neural network for input X, L is the total number of layers in the network, $f^{(l)}$ represents the transformation or mapping performed by layer l and $\Theta^{(l)}$ represents the parameters (weights and biases) of layer l.

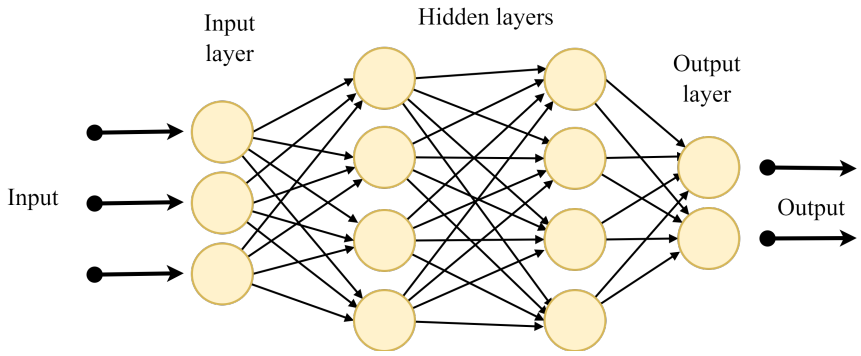


Figure 4.1: Visualization of a neural network composed of an input layer, an output layer, and two hidden layers in between.

Each layer l applies a transformation to its input, which is passed through an activation function $\sigma^{(l)}$ to introduce non-linearity:

$$f^{(l)}(a^{(l-1)}; \Theta^{(l)}) = \sigma^{(l)}(W^{(l)}a^{(l-1)} + b^{(l)})$$

Where $a^{(l-1)}$ is the output of the previous layer, $W^{(l)}$ is the weight matrix for layer l, $b^{(l)}$ is the bias vector for layer l, $\sigma^{(l)}$ is the activation function for layer l. Figure 4.1 depicts a Deep Neural Network - DNN where the hidden layers are all feedforward neural networks.

DL involves learning the optimal values of $\Theta^{(l)}$ through optimization techniques (e.g. Gradient descent) to minimize a loss function J that quantifies the error between the predicted output and the actual target values. As a result of their multilayered architecture and sophisticated transformation, DL models excel at spotting complex patterns in raw data, in contrast to conventional methods which may struggle to do so. There are some types of neural networks that can also be used to learn features depending on the specific task, including Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM).

CNN

If feedforward neural networks are replaced by convolutional layers, we have the CNN model. Suppose we have the input data as a 3D tensor X with dimensions $W \times H \times D$, where W is the width, H is the height, and D is the number of channels (e.g., three for RGB images). Given an input tensor X and a learnable filter (also known as kernel) K with dimensions $F \times F \times D$, where F is the filter size, we perform the convolution operation to obtain a feature map (also known as the activation map) A. Mathematically, the convolution operation can be defined as follows:

$$A_{i,j} = \sum_{m=0}^{F-1} \sum_{n=0}^{F-1} \sum_{d=0}^{D-1} X_{i+m,j+n,d} \cdot K_{m,n,d}$$

where $A_{i,j}$ is the value at position (i, j) in the feature map, $X_{i+m,j+n,d}$ is the value of the

input data at position $(i+m, j+n)$ in channel d, $K_{m,n,d}$ is the weight of the filter at position (m,n) in channel d. This convolution process computes the dot product between the filter K and the equally sized region in X centered at position (i,j). This sliding filter operation is repeated at every position in X to compute the full feature map A. Then, an activation function σ is applied element-wise to introduce non-linearity after a bias term b adding to each element of the feature map:

$$Y_{i,j} = \sigma(A_{i,j} + b).$$

Finally, to reduce the spatial dimensions of the feature maps, CNN often includes pooling layers. A common pooling operation is max-pooling, where we take the maximum value from a small region of the feature map. If we denote the pooling size as P, the operation can be defined as:

$$Y_{i,j} = \max_{m=0}^{P-1} \max_{n=0}^{P-1} Z_{i \cdot P+m, j \cdot P+n}.$$

Through training, the network learns the optimal values for the convolutional filter weights, biases, and fully connected layer weights to perform tasks like image classification, object detection, and more. The backpropagation algorithm is used to update these weights during training, optimizing the network's performance. The operation of a typical CNN is demonstrated in Figure 4.2.

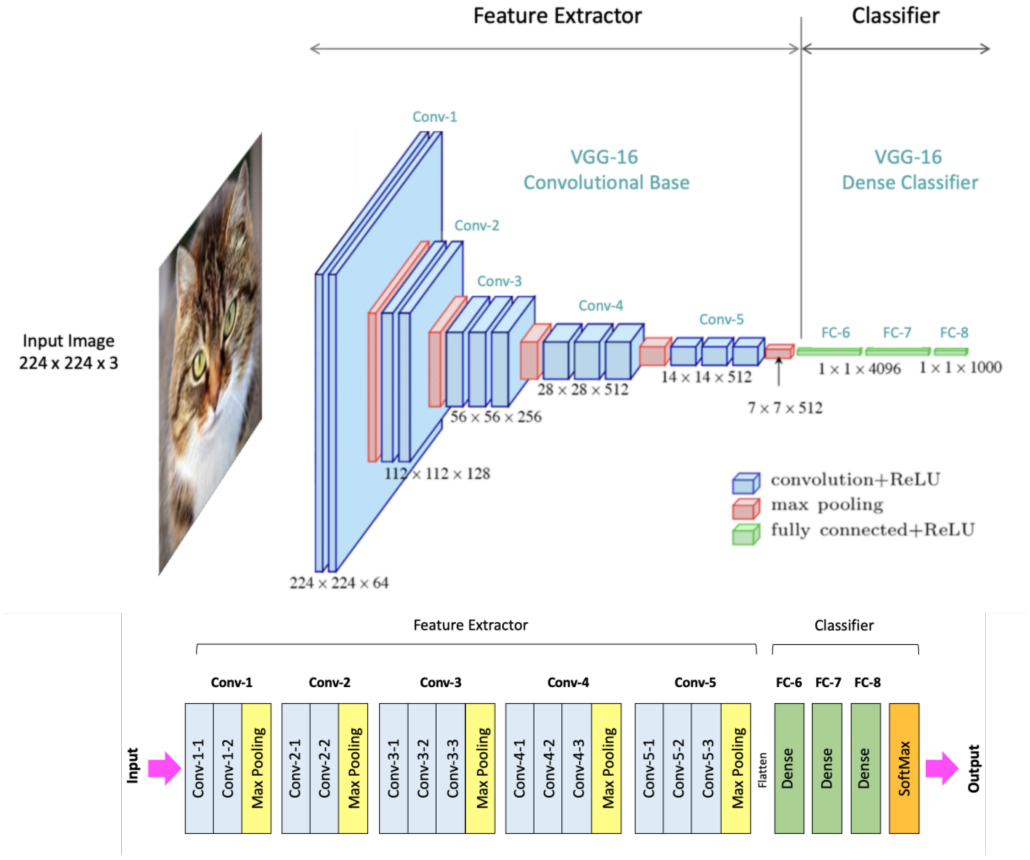


Figure 4.2: A convolutional neural network (CNN) architecture, (Extracted from [22]).

LSTM

LSTM (Long Short-Term Memory) is a type of RNN (Recurrent Neural Network) that can learn long-term dependencies. Given an input sequence $(X = (x_1, x_2, ..., x_T))$ with T time steps, an LSTM contains a memory cell (c_t) to remember values over long periods of time and three gates to control the flow of information: forget gate (f_t), input gate (i_t) and output gate (o_t). The LSTM standard architecture is demonstrated in Figure 4.3.

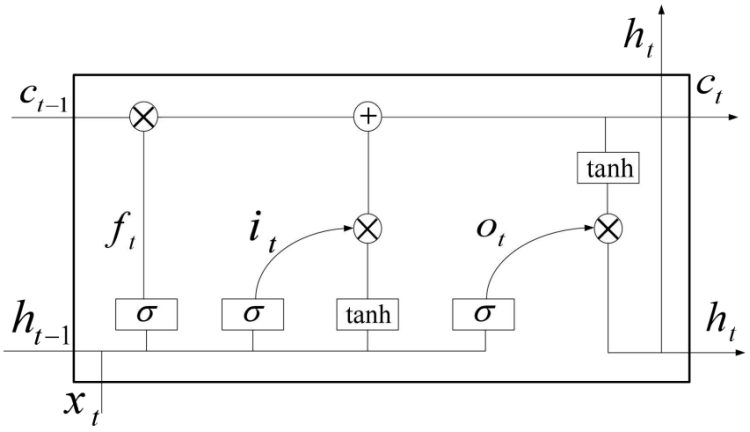


Figure 4.3: LSTM standard architecture.

At each time step t, the LSTM cell performs the following operations:

- Forget Gate: $(f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f))$
- Input Gate: $(i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i))$
- Memory Cell Update: $(c_t = f_t * c_{t-1} + i_t * \tanh(W_c \cdot [h_{t-1}, x_t] + b_c))$
- Output Gate: $(o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o))$
- Hidden State: $(h_t = o_t * \tanh(c_t))$

c_t, which is determined by the *input gate*, *input vector*, and *forget gate*, is the state of the memory cell at time step t. The final output of the LSTM is h_t, and its size is determined by the number of hidden layer nodes H. $sigma$ is the sigmoid function. The network parameters are denoted by W and b. Through training, the LSTM learns to adaptively capture dependencies of different time scales in sequence data.

Representation learning

Deep neural network (E.g. CNN, LSTM, Autoencoders, GANs, Transformers, etc.) has been widely proven the ability to learn features from raw data. The process of learning transformations or mappings of the data that capture the underlying factors of variations as well as semantic relationships between samples for ML tasks is called *representation learning*. According to Bengio et al. [9], representation learning offers a distinct advantage in knowledge transfer between tasks due to its capacity to capture high-level features and encourage *feature*

[22] learnopencv:`https://learnopencv.com/understanding-convolutional-neural-networks-cnn/`

re-use. Figure 4.4 visualizes how a DL model (CNN) learns features from the basic level to the high level.

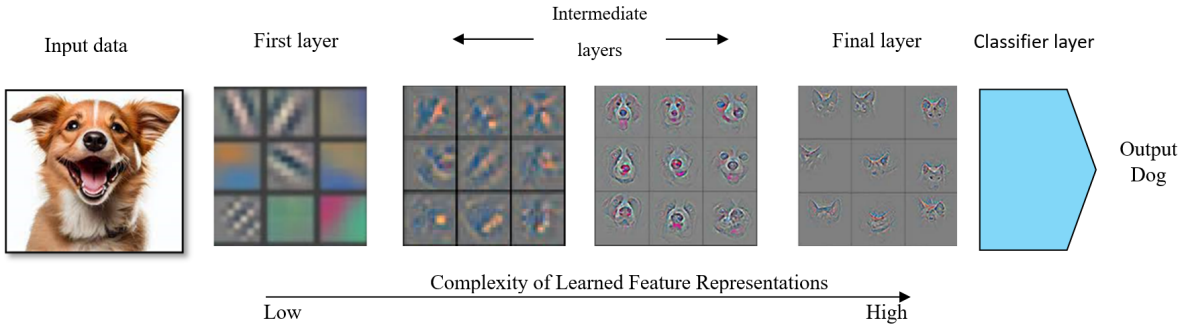


Figure 4.4: A convolutional neural network (CNN) acquires basic features in the first layers, progresses to complex intermediate layers, and results in sophisticated high-level feature representations that are then used for classification. (Modified from [104])

As a result, instead of extracting data features from scratch at each layer or processing step, neural networks can reuse the features they have learned from previous layers. This helps save time and enhances learning efficiency. Deep transfer learning adopts this idea. On one hand, the traditional supervised ML paradigm is built on learning in isolation, utilizing a single predictive model for a given task and a single dataset. This approach necessitates a substantial quantity of training instances and delivers optimal results when applied to well-defined and specific tasks. On the other hand, deep transfer learning applies the knowledge gained by a pre-trained DL model to one task and applies it to a new, related task, which helps to train a model from scratch. Typically, pre-trained language models use massive amounts of text data for the pre-training process (e.g. millions of words in the case of Word2Vec or billions of words in BERT). This allows the pre-trained language model to be adapted and specialized for a new task, rather than learning from the beginning. The features and patterns learned on the original task act as a starting point to learn the new task much faster and with less data. This makes deep transfer learning highly effective for real-world applications where large datasets are not readily available for every new problem.

4.2.2 Pre-trained language models

While transfer learning can be classified into various categories, in this study, we only consider the form of deep transfer learning called pre-train language models (aka. Large Language Models - LLMs for cases where the models are very large), which gain from the representation learning process of a large text dataset (aka. corpus) on sophisticated DL architectures. Three important pre-trained language models: Word2Vec, BERT, and RoBERTa, will be used in our study. As we only employ these models to extract feature-based representations for evaluating our proposed approaches and do not change the original structure of the models, we refrain from

delving into the technical details of each model. Details regarding the techniques employed are described in the cited publications.

Word2vec

Word2vec is an NLP approach that was published for the first time in 2013 by Mikolov et al. [60]. The word2vec algorithm utilizes a neural network model (Figure 1.5) to learn word associations from a huge corpus of text, called word embeddings. These embeddings are designed to capture the meaning and context of words in a given corpus, and they can be utilized for a range of NLP tasks like language translation, text classification, and information retrieval.

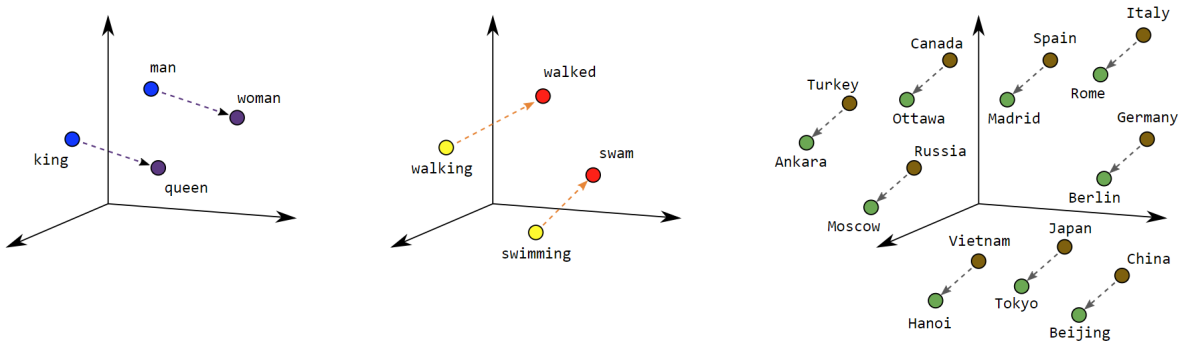


Figure 4.5: Word2Vec is capable of capturing multiple levels of similarity between words, allowing semantic and syntactic patterns to be reproduced using numerical vectors. For example, in the pairs of the words *"man"*, *"woman"* and *"king"*, and *"queen"*, two of them refer to males and two to females (extracted from [23]).

The primary approach of Word2Vec is to learn a high-dimensional vector representation of each word in a vocabulary, such that similar words are represented by similar vectors (Figure 4.5). This is accomplished by employing shallow neural networks such as a continuous bag-of-words (CBOW) or skip-gram model. The CBOW model learns to predict a target word based on the surrounding context, whereas the skip-gram model learns to predict the surrounding context words based on a target word. Both models are trained on a huge corpus of text, and the learned embeddings are then fed into other neural network models for downstream tasks. Despite its popularity and success, there are also some limitations to Word2Vec. One limitation is that the model uses a shallow neural network architecture, which might not be sufficient for capturing more complex word relationships. Another limitation is that it is dependent on a large corpus of text data, which is not always available or practical to obtain. Some alternatives and extensions to Word2Vec have been proposed in recent years, such as GloVe, FastText, and ELMO, which are also based on neural network architectures and aim to improve on some

[23] developers.google.com:`https://developers.google.com/machine-learning/crash-course/embeddings/translating-to-a-lower-dimensional-space`

of Word2Vec's limitations. Despite its limitations, Word2Vec is still widely used in NLP and remains an important tool for researchers and practitioners.

BERT

BERT (Bidirectional Encoder Representations from Transformers) is a transformer-based ML technique for natural language processing pre-training developed by Devlin et al. [24]. BERT is built on the transformer architecture, which processes input sequences using self-attention mechanisms [94]. The model is trained on a large corpus of text using a technique known as masked language modeling, with the objective of predicting a randomly masked word given its context. BERT also employs an additional task known as next-sentence prediction in which the model is trained to predict whether or not two sentences are consecutive. The bidirectional nature of BERT, which enables the model to see the context from both the left and right sides of the input, enables it to better comprehend the meaning of a given word and how it relates to other words. This has been shown to be particularly effective for tasks that require understanding the context-dependent meaning of a word, such as question answering and natural language inference. BERT has been shown to achieve state-of-the-art performance on a variety of benchmarks for natural language understanding, including GLUE and SQuAD [24]. It has also been implemented in numerous applications, including chatbots, search engines, summarizing lecture [28, 61, 68]. The architecture of BERT is depicted in Figure 1.6.

RoBERTa

RoBERTa (Robustly Optimized BERT Pre-training approach) was introduced in 2019 by Liu et al. [51] as a variation of the BERT model. It builds upon the pre-training approach of BERT by making several modifications to the training process, including increasing the size of the training dataset as well as the number of steps in the training process. RoBERTa only uses the masked language modeling objective and removes the next sentence prediction objective used in BERT. Additionally, instead of the 10% used in BERT, it uses a dynamic masking pattern where 15% of the tokens are replaced with a [MASK] token. Additionally, RoBERTa employs a larger batch size and a longer training schedule to enhance the model's learning capabilities. Similarly to the BERT pre-trained model, the RoBERTa pre-trained model can be fine-tuned on a variety of natural language understanding tasks, including question answering, sentiment analysis, and named entity recognition, by adding a task-specific layer on top of the pre-trained model and training it on the target task. It has been demonstrated that the enhancements to RoBERTa's pre-training procedure improved the model's performance on a wide variety of natural language understanding tasks. In particular, it has been demonstrated to outperform BERT and other state-of-the-art models on the GLUE and SQuAD benchmarks and to produce more consistent performance across a variety of fine-tuning tasks.

4.3 Our methods

4.3.1 Markov Chain Stochastic DCA as DL optimizer

Luu et al. [55, 56] proposed a new stochastic algorithm called Markov chain stochastic DCA (MCSDCA) to handle a class of stochastic (nonsmooth) DC programs with endogenous uncertainty in the absence of i.i.d. samples for training deep neural networks. This approach to DL is completely novel and extremely promising. The authors have demonstrated that the MCSDCA model outperforms existing approaches. In their research, however, only classical DL models were applied to time series prediction problems.

Through experimentation, the authors have demonstrated that the MCSDCA underdamped Langevin dynamics (MCSDCA-udLD) algorithm exhibits superior performance compared to the MCSDCA overdamped Langevin dynamics (MCSDCA-odLD) algorithm. In this study, we employed the MCSDCA-udLD algorithm for all experiments. The MCSDCA-udLD algorithm is presented from the study [55] as follows:

Algorithm 4 MCSDCA underdamped Langevin dynamics

Initialization. A starting point x^0, a sequence of positive numbers $\{\gamma_k\}$, a sequence of Markov chains' length $\{n_k\}$, the number of burn-in samples b, $\epsilon > 0$, $\delta > 0$, $t > 0$, set $k = 0$.

repeat

Set the starting state of the Markov chain $x_0^k := x^k$, and $v_0^k := 0$.

for $i = 0$ to $n_k - 2$ **do**

1. Receive a minibatch of data $D_{\text{minibatch}}$.
2. Compute a stochastic gradient $\tilde{\nabla} f(x_i^k)$ of f using $D_{\text{minibatch}}$.
3. Compute the conditional expectation of v_{i+1}^k and x_{i+1}^k given x_i^k and v_i^k as follows

$$\mathbb{E}(v_{i+1}^k) = e^{-2\delta} v_i^k - \frac{1}{2}(1 - e^{-2\delta}) \left(\tilde{\nabla} f(x_i^k) + \frac{1}{t}(x_i^k - x^k) \right),$$

$$\mathbb{E}(x_{i+1}^k) = x_i^k + \frac{1}{2}(1 - e^{-2\delta}) v_i^k - \frac{1}{2} \left(\delta - \frac{1}{2}(1 - e^{-2\delta}) \right) \left(\tilde{\nabla} f(x_i^k) + \frac{1}{t}(x_i^k - x^k) \right).$$

4. Sample $v_{i+1}^k \sim \mathbb{E}(v_{i+1}^k) + \sqrt{c_2}\mathcal{N}(0, I)$.
5. Sample $x_{i+1}^k | v_{i+1}^k \sim \mathbb{E}(x_{i+1}^k) + c_3 c_2^{-1}(v_{i+1}^k - \mathbb{E}(v_{i+1}^k)) + \sqrt{(c_1 - c_3^2 c_2^{-1})}\mathcal{N}(0, I)$.

end for

Compute $y^k = \dfrac{1}{n_k - b} \sum_{i=b}^{n_k - 1} x_i^k$.

Solve the following convex problem,

$$x^{k+1} = \arg\min_x \left\{ \frac{1}{2t}\|x\|^2 - \frac{1}{t}\langle x, y^k \rangle + \frac{\gamma_k}{2}\|x - x^k\|^2 \right\}, \tag{4.1}$$

which has the closed-form solution

$$x^{k+1} = \frac{t\gamma_k}{1 + t\gamma_k} x^k + \frac{1}{1 + t\gamma_k} y^k.$$

$k = k + 1$.

until Stopping criterion

In this chapter, our emphasis will be on conducting a thorough assessment of this new optimizer model concerning its application in advanced DL frameworks within the context of Deep Natural Language Processing for Healthcare. On the one hand, we will compare our proposed architecture, with traditional DL models, including CNN and LSTM. On the other hand, MCSDCA will serve as the DL optimizer for comparison with established optimizers like Gradient Descent and ADAM across various DL architectures employing diverse pre-trained language models.

4.3.2 Hybrid CNN - BiLSTM architecture

CNN and LSTM models have been widely applied to NLP applications and text classification, in particular [53, 96, 105, 110]. CNN is a powerful algorithm for NLP tasks, as it can extract

local features from the individual words or sub-word units (such as n-grams), that can be used to make predictions or perform other tasks. On the other hand, LSTM is particularly good at handling sequential information, such as the order of words in a sentence. LSTM contains memory cells that can store information from previous time steps and use it to manage long-term dependencies, allowing them to comprehend relationships between words that are far apart in the input text. This is especially useful for NLP tasks in which the meaning of a word is dependent on its context. CNN-LSTM is a robust architecture that can be applied to a variety of tasks and can significantly improve performance when compared to CNN or LSTM alone. There are some variations of hybrid CNN-LSTM have been introduced, such as [36, 53, 105].

Combining the CNN and LSTM in a single model is a promising strategy for training. The main advantage of this architecture is that it allows the model to consider both local and sequential information in the input data, as a result, the model can learn more complex relationships between words and comprehend the text's meaning more precisely. BiLSTM, a variant of LSTM, is proven to outperform the regular unidirectional LSTM [88] thanks to the ability to capture the context of sequence input in two directions, so we use BiLSTM in our proposed method. Thus, our proposed approach uses a pre-trained language model (Word2Vec, BERT, RoBERTa) to extract initial features from the input. These embedding features are then fed into a hybrid CNN-BiLSTM architecture, which acts as a further refined feature extractor. The refined features from the CNN-BiLSTM are then passed through the classification layers (fully connected layers) for predicting the output labels. Figure 4.6 illustrates how we combine CNN with BiLSTM to form a hybrid model.

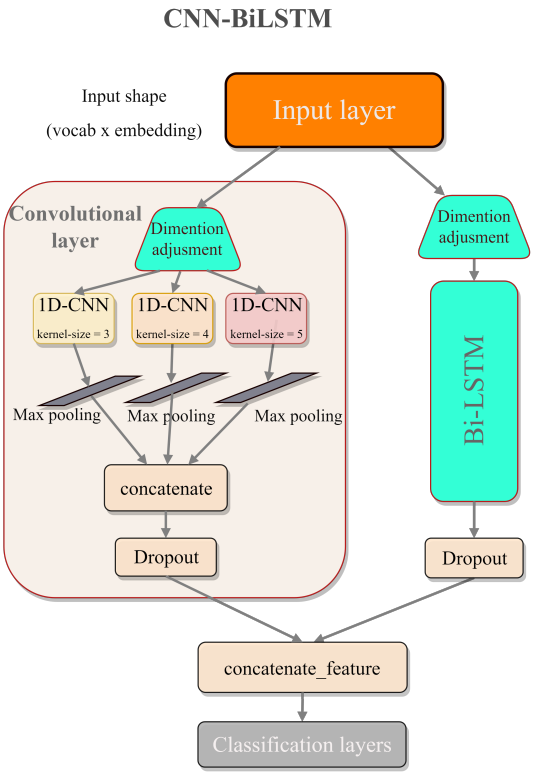


Figure 4.6: The proposed CNN-BiLSTM model architecture diagram.

4.3.3 Hybrid CNN-BiLSTM with Word2Vec embedding

Word2vec layer: With the Word2vec layer, the raw text is transformed into vector operations in the w-dimensional vector space through training.

The architecture of the CNN-BiLSTM with Word2Vec embedding is in Figure 4.7.

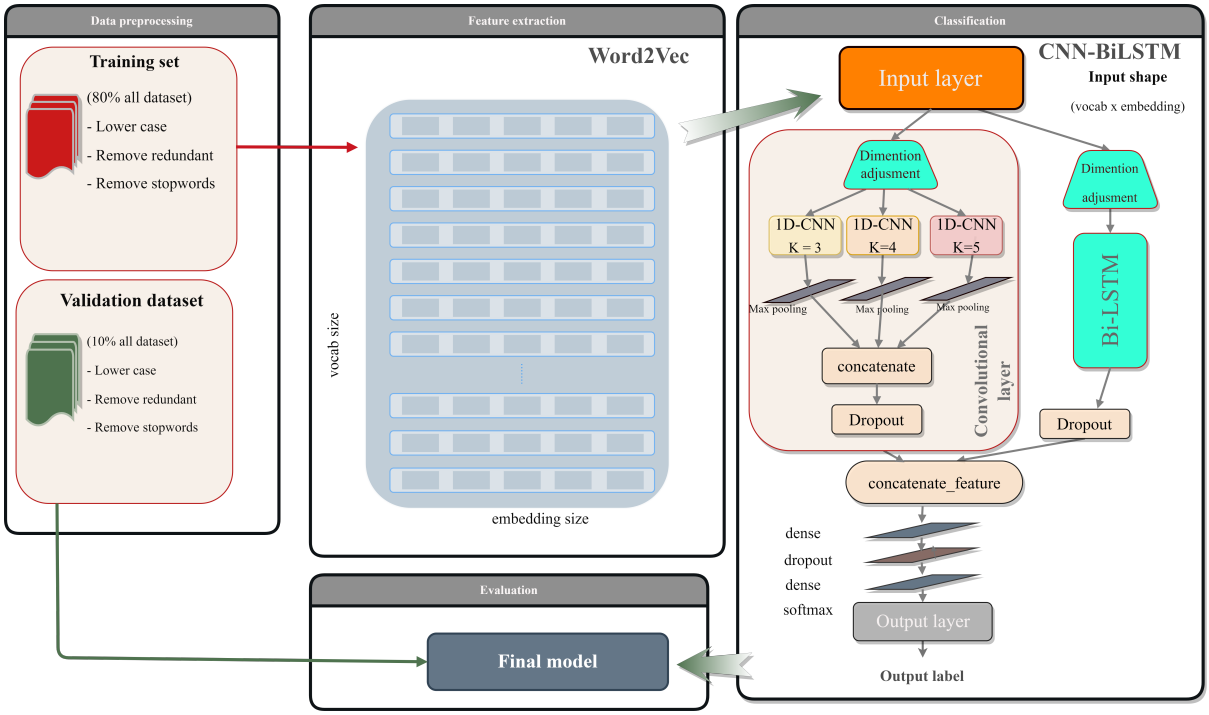


Figure 4.7: The proposed CNN-BiLSTM model architecture with Word2Vec embedding.

4.3.4 Hybrid CNN-BiLSTM with BERT embedding

BERT embedding layer: BERT is a pretrained model that requires input text to be in a specific format. In BERT, text is transformed into numerical vectors through tokenization and embedding. Tokenization involves splitting the input text into individual tokens, such as words or subwords, and assigning them unique indices. These indices are then used as input to the BERT model. Embedding involves mapping each token index to a dense vector, called an embedding vector, which represents the token's meaning in a continuous vector space. These embedding vectors are learned during training and capture the contextual relationships between words in the input text.

The architecture of the CNN-BiLSTM with BERT embedding is in Figure 4.8.

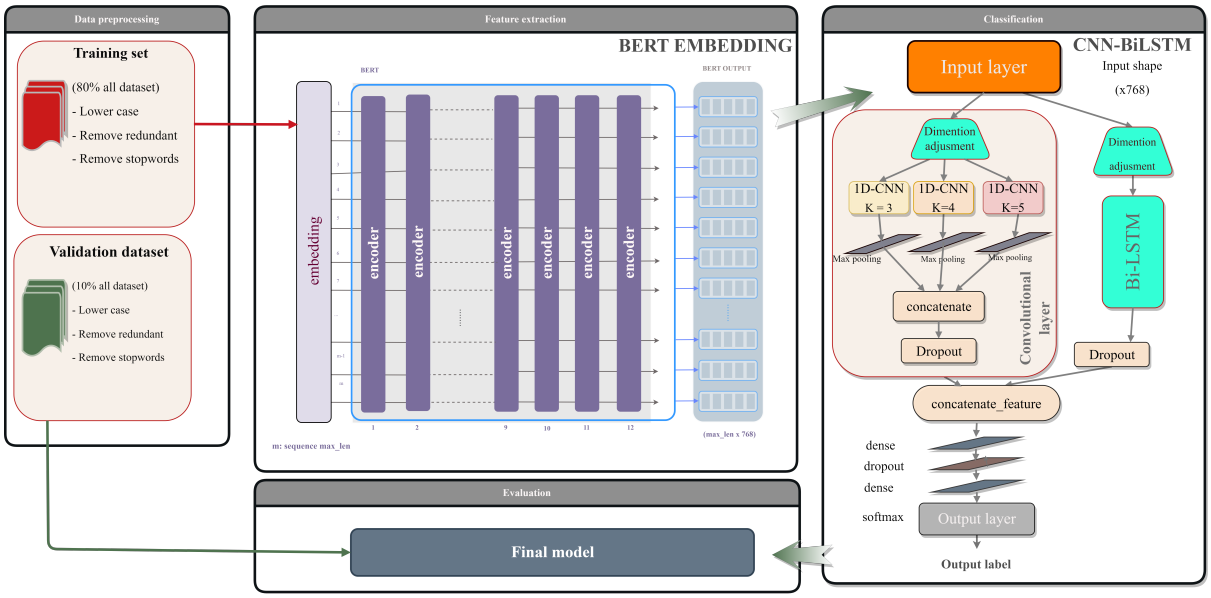


Figure 4.8: Proposed CNN-BiLSTM model architecture with BERT embedding.

4.3.5 Hybrid CNN-BiLSTM with RoBERTa embedding

RoBERTa embedding layer: As a variant of BERT, RoBERTa is a transformer-based language model that processes input sequences and generates contextualized representations of words in a sentence using self-attention. The larger dataset and a more effective training technique distinguish RoBERTa from BERT. RoBERTa also uses dynamic masking during training to develop more robust and generalizable word representations. The architecture of the CNN-BiLSTM with RoBERTa embedding is in Figure 4.9.

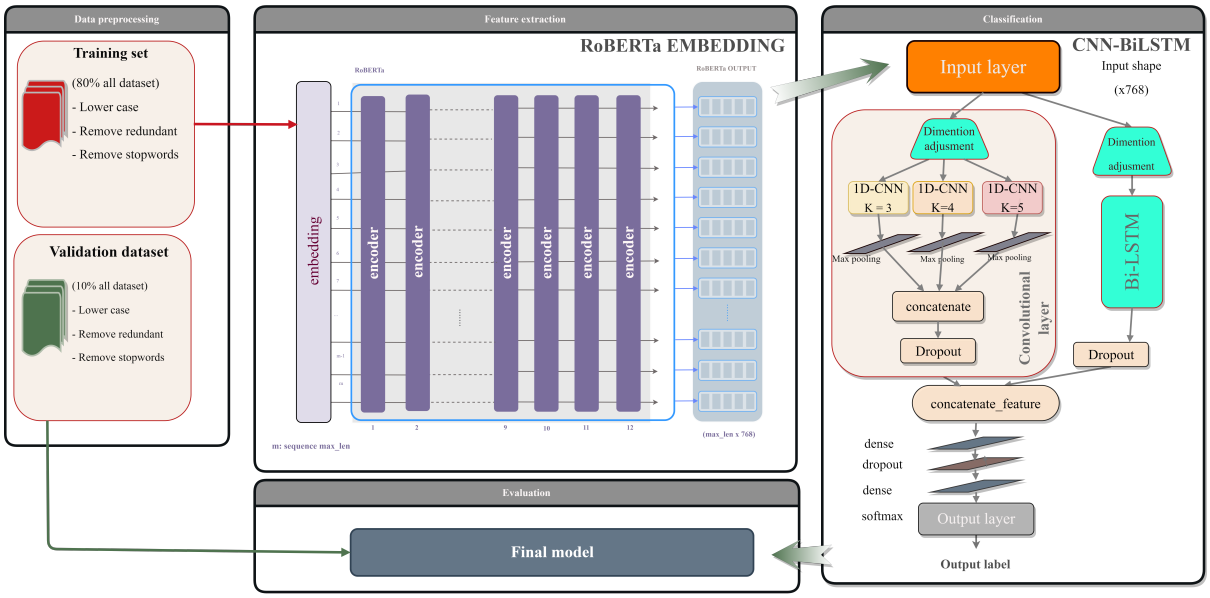


Figure 4.9: The proposed CNN-BiLSTM model architecture with RoBERTa embedding.

4.4 Healthcare text classification problems and Datasets

In the medical and healthcare industry, textual information exists in a variety of formats, including electronic health records (EHRs). This type of data is frequently processed using natural language processing (NLP) methods. NLP (often associated with ML techniques) illustrates how artificial intelligence collects and evaluates unstructured data from human language to extract patterns, meaning, and then provide feedback. As a result, the healthcare sector is able to make the best use possible of unstructured data, including electronic clinical notes, discharge summaries, and other kinds of information. NLP combined with ML techniques can assist physicians in detecting various diseases from electronic health records, allowing managers at medical facilities to automate management workflows, spend more time on patient care, and enhance the patient experience using real-time data. This chapter will employ three healthcare datasets in the following three demanding healthcare problems: *1.* Predict hallmarks of cancer, *2.* Extracting actions for physicians from hospital discharge notes, *3.* Classification of medical specialties from medical notes. The details of the datasets are shown in table 4.1.

4.4.1 Hallmarks of Cancer (HoC)

Created by Baker et al. at 2016 [6], the Hallmarks of Cancer (HoC) contains 1852 abstracts from PubMed publications that have been manually annotated by subject matter experts. The annotations were done using a hierarchical taxonomy containing 37 different classes. Each sentence in the corpus may be assigned zero or more labels from this taxonomy. The dataset

is for multi-class classification problems (10 classes). In this study, we only use one label: *1. Sustaining proliferative signaling (PS)* and transform it into a binary classification problem.

4.4.2 Extracting action items for physicians from hospital discharge notes (CLIP)

CLIP [64] was built using the well-known MIMIC-III clinical dataset. This dataset poses a multi-label classification problem, with labels being assigned at the sentence level. In total, there are 107,494 sentences in the dataset after tokenization using the provided sentence tokenizer. Of these sentences, 12,079 have been assigned at least one label from the predefined set of labels for the multi-label classification task. The sourcecode for processing the data is in the GitHub link [24]. The dataset is also for multi-class classification problems (7 labels). In this study, to be simple for our text classification problem, we only use one label: *Appointment* and transform it into a binary classification problem.

4.4.3 Medical transcriptions (MedTrans)

This dataset is public on the Kagglle website[25]. Obtaining medical data can be exceedingly challenging due to the stringent privacy regulations imposed by HIPAA (Health Insurance Portability and Accountability Act). To address this issue, this dataset offers a valuable solution by making available a collection of medical transcription samples which are scraped from scraped from mtsamples.com. The original dataset is designed for a text classification problem with 40 labels, each representing a medical specialty, such as Surgery, Radiology, Gastroenterology, and so forth. The number of instances per label varies significantly, ranging from thousands to just a few dozen instances. To simplify the dataset, all labels with fewer than 200 instances have been grouped together under a single label named "other." As a result, the classification problem has been transformed into one with a total of 9 labels.

4.4.4 Datasets' information

dataset	name	#trainset	#valset	#testset	#total
HoC_1	Hallmarks of cancer	1,482	185	185	1,852
CLIP_1	CLIP	86,280	10,786	10,785	107,851
MedTrans	Medical Transcriptions	3,118	390	390	3,898

Table 4.1: Information of the datasets (Number of the trainset, valset, testset and total number of samples in our experiments.

[24] https://github.com/asappresearch/clip
[25] https://www.kaggle.com/datasets/tboyle10/medicaltranscriptions

4.5 Experiment setting

4.5.1 Comparative algorithms and optimizers

We compare the proposed neural network architecture (CNN-BiLSTM) with the two following baselines:

- CNN: the single CNN branch of 4.6 (On the right) without LSTM. This model includes three 1D-convolutional layers with different kernel sizes (K = 3,4,5) and is followed by Pooling layers. Then, a Dropout layer is added as a regularization layer to avoid overfitting. Finally, all three results are concatenated as features for classification.
- BiLSTM: the single LSTM branch of 4.6 (On the left) without CNN. This model includes one BiLSTM layer followed by a Dropout layer. The features from the BiLSTM layer are then ready for classification at the next step.

Combination with three pre-trained language models: Word2Vec, BERT, RoBERTa with MCSDCA, we compare our methods with 2 optimizers: Stochastic Gradient Descent (SGD) and Adam algorithms to comprehensively evaluate the efficacy of MCSDCA on advanced DL achitectures.

The goal of our experiments on two important optimizers: SGD[26] and ADAM[27] is to comprehensively evaluate the efficacy of MCSDCA in combination with these advanced DL architectures and optimization algorithms.

4.5.2 Experiment setup and model setting

In order to implement our proposed method CNN-BiLSTM implementations, we used Python 3 and the Pytorch 1.13.1[28] package as a DL framework. Pre-trained language models: BERT[29], RoBERTa[30] is from HuggingFace library. Word2Vec is from the popular Github source[31]. The training procedure is sped up by using a server with a graphic processing unit (GPU) and CUDA (10.1) kernel. As a detail, the server computer DELL Precision 7820 Tower with a CPU of Intel Xeon(R) Gold 5118 2.30GHz 2.29GHz (2 processors), 128 GB of memory (RAM), and a GeForce RTX 1080, 8GB has been used to implement the training procedure.

Parameters of DL models:

- Setting for optimizers: For the baseline algorithms (Adam, SGD), we use the default values of SGD (lr=1e-2,) and ADAM (lr = 1e-3, weight decays = 0.0, exponential decay

[26]https://pytorch.org/docs/stable/generated/torch.optim.SGD.html
[27]https://pytorch.org/docs/stable/generated/torch.optim.Adam.html?highlight=adam
[28]https://pytorch.org/
[29]https://huggingface.co/transformers/v3.0.2/model_doc/bert.html
[30]https://huggingface.co/docs/transformers/model_doc/roberta
[31]https://github.com/mmihaltz/word2vec-GoogleNews-vectors

rates: $\beta 1$ = 0.9, $\beta 2$= 0.999, eps=1e-08)) as described in Pytorch 1.13.1. For MCSDCA, we set the values as the author set in their experiments in [56]. Thus, we set time $1/t = 10^{-4} \cdot 1.001$ and $\epsilon = 10^{-8}$. The Langevin step count is denoted as n_k and is determined by the formula $n_k = n_{\text{init}} + \lfloor k^{\lambda} \rfloor$, where n_{init} is set to 20, λ is equal to 0.1, and k is specifically equal to 1. The step size of the Langevin dynamics is set at 0.001.

- Setting for CNN-BiLSTM: The CNN layer size is (D, Fz) where D is the embedding size and Fz is the out channels of CNN, here we set Fz = 100. The kernel number of the CNN layers is 3 with kernel sizes in (3, 4, 5). The number of LSTM layers is one with layer size $(D, \frac{D}{2})$. For the classification layers, the size of the first fully connected layer is set $(L, \frac{L}{2})$, the size of the second fully connected layer is set $(\frac{L}{2}, C)$ with C is number classes and L is the concatenated feature size of the CNN and BiLSTM layers, $L = 3 * Fz + \frac{D}{2}$. All dropout layers are set with a value of 0.4.

- Setting for pre-trained language models: Due to hardware constraints (8GB GPU), we are limited to using smaller versions of BERT and RoBERTa, specifically *bert-base-uncased* and *roberta-base*. The default embedding size for these models is set at 768. Both of these pre-trained models are loaded directly from the HuggingFace *transformers* package[32]. Furthermore, our tokenization process aligns with the established techniques employed for BERT and RoBERTa within the *transformers* package. The Word2Vec embedding size remains at its default setting of 300 dimensions.

4.5.3 Evaluation metrics

Two measurement metrics, including Accuracy and F1-Score, have been used to assess the predictive performance of our models. Using TP to represent True Positive, TN for True Negative, FP for False Positive, and FN for False Negative, Accuracy is computed as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}$$

To thoroughly evaluate the performance of a model, especially in cases where the dataset is imbalanced in healthcare problems, we must consider both precision and recall. The F1 score is a useful metric that takes into account both of them.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{F1 Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{\text{TP}}{\text{TP} + \frac{1}{2}(\text{FP} + \text{FN})}$$

[32]https://huggingface.co/docs/transformers/index

F1-score is referred to as the harmonic mean of precision and recall for a more balanced summarization of model performance.

4.6 Numerical experiment and Discussion

4.6.1 Experiment 1: Comparison of CNN, LSTM and the proposed CNN-BiLSTM with Word2Vec

Dataset	Metric	Accuracy			F1-Score		
		SGD	ADAM	MCSDCA	SGD	ADAM	MCSDCA
HoC_1	CNN-Word2Vec	84.95	84.95	**93.55**	45.93	45.93	**86.14**
	BiLSTM-Word2Vec	84.94	93.01	**93.54**	45.93	**87.88**	87.74
	CNN-BiLSTM-Word2Vec	84.90	94.60	**95.16**	45.93	89.79	**89.85**
CLIP_1	CNN-Word2Vec	95.89	98.58	**98.67**	48.95	89.91	**90.71**
	BiLSTM-Word2Vec	**98.76**	98.59	98.75	**92.15**	90.75	91.94
	CNN-BiLSTM-Word2Vec	98.62	98.65	**98.82**	90.63	91.39	**92.16**
MedTrans	CNN-Word2Vec	96.32	**96.55**	96.50	95.36	**95.44**	95.42
	BiLSTM-Word2Vec	96.55	**96.73**	96.62	96.08	**96.23**	96.15
	CNN-BiLSTM-Word2Vec	97.12	97.32	**97.35**	96.89	97.05	**97.13**

Table 4.2: Results of the proposed CNN-BiLSTM architecture and traditional methods with Word2Vec embedding on the 3 healthcare datasets.

Comments on the experimental result: The experimental results presented in Table 4.2 compare the performance of the proposed CNN-BiLSTM architecture with traditional methods, all utilizing Word2Vec embeddings, across three healthcare datasets: HoC, CLIP, and MedTrans. Notably, the CNN-BiLSTM architecture consistently outperforms CNN and BiLSTM in terms of accuracy and F1-Score. In the HoC dataset, the CNN-BiLSTM-Word2Vec model stands out with an accuracy of 95.16% and an F1-Score of 89.85%. For CLIP, the model also excels, achieving an accuracy of 98.82% and an F1-Score of 92.16%. In the MedTrans dataset, the CNN-BiLSTM-Word2Vec model maintains strong performance, achieving an accuracy of 97.35% and an F1-Score of 97.13%. In the comparison of optimizers, while SGD is unstable and performs poorly in some circumstances, ADAM and MCSDCA perform better on these datasets, particularly in terms of F1-Score results. These results demonstrate the superiority of the proposed CNN-BiLSTM architecture, particularly when coupled with MCSDCA, in the context of healthcare datasets, offering promising avenues for further exploration in machine learning and optimization.

4.6.2 Experiment 2: Comparison of pre-trained language models

Dataset	Model	Accuracy			F1-Score		
		SGD	ADAM	MCSDCA	SGD	ADAM	MCSDCA
HoC_1	CNN-BiLSTM-Word2Vec	84.90	94.60	**95.16**	45.93	89.79	**89.85**
	CNN-BiLSTM-BERT	96.12	96.85	**97.24**	91.87	92.35	**93.33**
	CNN-BiLSTM-RoBERTa	96.73	97.18	**97.55**	91.95	92.47	**93.51**
CLIP_1	CNN-BiLSTM-Word2Vec	98.62	98.65	**98.82**	90.63	91.39	**92.16**
	CNN-BiLSTM-BERT	98.79	98.85	**99.05**	91.15	91.38	**92.37**
	CNN-BiLSTM-RoBERTa	98.82	98.95	**99.20**	91.21	91.42	**92.43**
MedTrans	CNN-BiLSTM-Word2Vec	97.12	97.32	**97.35**	96.89	97.05	**97.13**
	CNN-BiLSTM-BERT	98.46	**98.97**	**98.97**	98.46	**98.94**	98.92
	CNN-BiLSTM-RoBERTa	98.97	**99.23**	99.20	98.94	**99.20**	99.14

Table 4.3: Result of CNN-BiLSTM with different pre-trained language models: Word2Vec, BERT and RoBERTa.

Comments on the experimental result: The experimental results presented in Table 4.3 showcase the performance of CNN-BiLSTM models employing various pre-trained language models, namely Word2Vec, BERT, and RoBERTa, across three datasets: HoC, CLIP, and MedTrans. In consistent the previous experiment result, the MCSDCA optimizer outperforms SGD and can be comparable with ADAM in terms of accuracy and F1-Score. When comparing pre-trained language models, it becomes evident that RoBERTa consistently delivers the most impressive performance across all scenarios. These findings underline the significance of the MCSDCA optimizer and highlight the effectiveness of RoBERTa in certain scenarios. RoBERTa's consistently impressive performance across various scenarios underscores the advantages of larger pre-trained language models (In terms of data training and model architecture). Without additional training data, in summary, leveraging a larger pre-trained language model with the same architecture typically results in improved performance.

4.7 Conclusion

In conclusion, this chapter has proposed a hybrid CNN-BiLSTM method incorporated with the MCSDCA algorithm and pre-trained language models in an effort to mitigate data scarcity in the domain of healthcare. Through the presentation of case studies, we have demonstrated the efficacy of our proposed approach in tasks such as cancer prediction, classification of specialties from medical text, and clinical action classification based on discharge summaries.

However, it's important to acknowledge certain limitations in our current study. Due to constraints of time, computational resources, and access to healthcare datasets, we were unable to conduct tests on a broader range of datasets. Moreover, we have utilized the feature-based method thus far. Notably, we have not yet incorporated techniques for fine-tuning our strategy.

The application of the fine-tuning technique to larger clinical data corpora in future research can enhance the performance of the proposed method. In addition, while our current research has yielded promising results, we look forward to addressing these limitations and leveraging the capabilities of state-of-the-art language models in our ongoing work to enhance the robustness and applicability of the proposed method in healthcare-related tasks.

Chapter 5

Conclusion and perspectives

This thesis focuses on solving three challenges in ML: large-scale data with kernel SVM, imbalanced data for finance applications, and scarcity of data in the domain of healthcare. The principal methodologies of this thesis are based on DC Programming and DCA, which are well-known powerful tools due to their effectiveness and efficiency for non-smooth and non-convex optimization problems.

In the first part, we have proposed a new block coordinate descent algorithm based on DCA called BC-DCASVM to address large-scale learning challenges with kernel SVM. By integrating DCA with a unified scheme and applying the block coordinate method, BC-DCASVM is capable of handling high-dimensional variables and large datasets efficiently. Specifically, at each iteration of the training process, the method reduces the objective value by optimizing one block of coordinates at a time while keeping other blocks fixed. Additionally, the problem to solve at each iteration is a closed-form system of linear equations. Consequently, calculation time can be considerably reduced while memory overflow is prevented, which is essential for computing in the context of big data. Moreover, with the unified schema, the proposed method is capable of handling various common loss functions in SVM, including convex and nonconvex losses, as well as classification and regression losses. Extensive experiments on real-world datasets demonstrate that BC-DCASVM consistently outperforms existing alternatives in terms of both accuracy and runtime. Notably, it solved high-dimensional problems much faster while maintaining competitive prediction performance.

To solve the imbalanced data problem in the context of finance, we propose the weighted bagging algorithms based on the DCA method with two versions, namely BaggingDCA and CSB-DCA. BaggingDCA is a technique that provides weights to each base model based on its performance. This is achieved through the use of LS-DC, a unified approach that can handle diverse loss functions in machine learning, regardless of whether they are convex or non-convex. We conducted an evaluation of the BaggingDCA algorithm on multiple benchmark datasets and subsequently compared its performance to that of existing bagging methods. CSB-DCA builds upon BaggingDCA by integrating cost-sensitive learning to assign higher weights to misclassified minority class instances, an important consideration for imbalanced financial problems

where errors in the rare class may carry substantial costs. CSB-DCA is tailored to address the challenges of imbalanced financial data classification. These proposed methods inherit the advantages of traditional bagging, such as reduced variance, parallel computation support, and the prevention of overfitting, while providing improved model accuracy. Experimental analysis on various benchmark and financial datasets, including those from UCI and Kaggle, as well as our own, validates their effectiveness in real-world scenarios, including fraud detection, credit approval, and bankruptcy prediction. While this research focused on bagging-based methods, future work could explore integrating the proposed weighting frameworks into other ensemble techniques like boosting, stacking, or pasting. Additionally, evaluating alternative base learners beyond traditional classifiers may provide additional performance gains.

In the last part, with an effort to reduce the effects of data scarcity in the domain of healthcare, we proposed a hybrid CNN-BiLSTM technique that incorporates the MCSDCA algorithm and powerful pre-trained language models. Combining CNN and LSTM into a singular training model is a compelling approach for training as it leverages the strengths of both architectures. The key benefit of this combined model is its ability to simultaneously consider local as well as sequential information within the input data. Consequently, this model can acquire a deeper understanding of intricate word relationships, leading to a more precise comprehension of textual content. Moreover, using the promising prospects of the Markov-chain stochastic DCA (MCSDCA) algorithm, an optimizer based on DCA for deep learning that has been evaluated on traditional deep learning architectures [55], we evaluate the capacity of the algorithm on advance deep learning architectures to address some significant challenges in the healthcare industry. Additionally, we leverage several pre-trained language models, inspired by transfer learning principles, to mitigate the data scarcity problem. Through comparisons with prevalent optimizers, deep learning models, and pre-trained language models, our approach shows competitiveness with current methods. We have proved the efficacy of our proposed approach in a variety of tasks, such as the classification of specialties from medical texts, the prediction of cancer from hallmarks of cancer, and clinical action classification based on discharge summaries. In future research, applying fine-tuning to larger clinical datasets in future studies could further improve the proposed method's performance. Additionally, while our current work has shown promising results, we aim to address existing limitations and utilize advanced language models in our ongoing research. The goal is to strengthen the proposed approach and make it more widely applicable to healthcare tasks.

Bibliography

[1] L. Adlung, Y. Cohen, U. Mor, and E. Elinav. Machine learning in clinical decision making. *Med*, 2(6):642–665, 2021.

[2] E. Alpaydin. *Introduction to machine learning*. MIT press, 2020.

[3] E. Alsentzer, J. R. Murphy, W. Boag, W.-H. Weng, D. Jin, T. Naumann, and M. McDermott. Publicly available clinical bert embeddings. *arXiv preprint arXiv:1904.03323*, 2019.

[4] N. Altman and M. Krzywinski. Ensemble methods: bagging and random forests. *Nature Methods*, 14(10):933–935, 2017.

[5] D. B. Araya, K. Grolinger, H. F. ElYamany, M. A. Capretz, and G. Bitsuamlak. An ensemble learning framework for anomaly detection in building energy consumption. *Energy and Buildings*, 144:191–206, 2017.

[6] S. Baker, I. Silins, Y. Guo, I. Ali, J. H"ogberg, U. Stenius, and A. Korhonen. Automatic semantic classification of scientific literature according to the hallmarks of cancer. *Bioinform.*, 32(3):432–440, 2016.

[7] A. Beck and L. Tetruashvili. On the convergence of block coordinate descent type methods. *SIAM journal on Optimization*, 23(4):2037–2060, 2013.

[8] R. Bekkerman, M. Bilenko, and J. Langford. *Scaling up machine learning: Parallel and distributed approaches*. Cambridge University Press, 2011.

[9] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[10] J. Błaszczyński and J. Stefanowski. Neighbourhood sampling in bagging for imbalanced data. *Neurocomputing*, 150:529–542, 2015.

[11] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of COLT '92*, pages 144–152, 1992.

[12] S. Brasil, C. Pascoal, R. Francisco, V. dos Reis Ferreira, P. A. Videira, and G. Valadão. Artificial intelligence (ai) in rare diseases: is the future brighter? *Genes*, 10(12):978, 2019.

[13] L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[14] L. Breiman. Stacked regressions. *Machine learning*, 24(1):49–64, 1996.

[15] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[16] I. Brown and C. Mues. An experimental comparison of classification algorithms for imbalanced credit scoring data sets. *Expert Systems with Applications*, 39(3):3446–3453, 2012.

[17] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011.

[18] Y.-C. Chang, K.-H. Chang, and G.-J. Wu. Application of extreme gradient boosting trees in the construction of credit risk assessment models for financial institutions. *Applied Soft Computing*, 73:914–920, 2018.

[19] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[20] S. Cheriyan, S. Ibrahim, S. Mohanan, and S. Treesa. Intelligent sales prediction using machine learning techniques. In *2018 International Conference on Computing, Electronics & Communications Engineering (iCCECE)*, pages 53–58. IEEE, 2018.

[21] H.-Y. Chou, P.-Y. Lin, and C.-J. Lin. Dual coordinate-descent methods for linear one-class svm and svdd. In *Proc.SIAM Int.Conf.Data Min.*, pages 181–189. SIAM, 2020.

[22] C. Cortes and V. Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, 1995.

[23] D. Devi, S. K. Biswas, and B. Purkayastha. A cost-sensitive weighted random forest technique for credit card fraud detection. In *2019 10th international conference on computing, communication and networking technologies (ICCCNT)*, pages 1–6. IEEE, 2019.

[24] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[25] C. Elkan. The foundations of cost-sensitive learning. In *IJCAI*, volume 17, pages 973–978. Lawrence Erlbaum Associates Ltd, 2001.

[26] R. P. Espíndola and N. F. Ebecken. On extending f-measure and g-mean metrics to multi-class problems. *WIT Transactions on Information and Communication Technologies*, 35:25–34, 2005.

[27] Y. Freund, R. E. Schapire, et al. Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156. Citeseer, 1996.

[28] J.-C. Gu, T. Li, Q. Liu, Z.-H. Ling, Z. Su, S. Wei, and X. Zhu. Speaker-aware bert for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2041–2044, 2020.

[29] M. Gurbuzbalaban, A. Ozdaglar, P. A. Parrilo, and N. Vanli. When cyclic coordinate descent outperforms randomized coordinate descent. *Advances in Neural Information Processing Systems*, 30, 2017.

[30] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing. Learning from class-imbalanced data: Review of methods and applications. *Expert systems with applications*, 73:220–239, 2017.

[31] A. Harnoune, M. Rhanoui, M. Mikram, S. Yousfi, Z. Elkaimbillah, and B. El Asri. Bert based clinical knowledge extraction for biomedical knowledge graph construction and analysis. *Computer Methods and Programs in Biomedicine Update*, 1:100042, 2021.

[32] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[33] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th international conference on Machine learning*, pages 408–415, 2008.

[34] Z. Huang, Z. Pan, and B. Lei. Transfer learning with deep convolutional neural network for sar target classification with limited labeled data. *Remote sensing*, 9(9):907, 2017.

[35] R. Ž. Jovanović, A. A. Sretenović, and B. D. Živković. Ensemble of various neural networks for prediction of heating energy consumption. *Energy and Buildings*, 94:189–199, 2015.

[36] F. Karim, S. Majumdar, H. Darabi, and S. Chen. Lstm fully convolutional networks for time series classification. *IEEE access*, 6:1662–1669, 2017.

[37] T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano. Comparing boosting and bagging techniques with noisy and imbalanced data. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 41(3):552–568, 2010.

[38] M.-J. Kim, D.-K. Kang, and H. B. Kim. Geometric mean based boosting algorithm with over-sampling to resolve data imbalance problem for bankruptcy prediction. *Expert Systems with Applications*, 42(3):1074–1082, 2015.

[39] H. M. Le, H. A. Le Thi, and M. C. Nguyen. Sparse semi-supervised support vector machines by DC programming and DCA. *Neurocomputing*, 153:62–76, 2015.

[40] H. A. Le Thi, H. M. Le, and T. P. Dinh. Feature selection in machine learning: an exact penalty approach using a difference of convex function algorithm. *Machine Learning*, 101(1):163–186, 2015.

[41] H. A. Le Thi, H. M. Le, V. V. Nguyen, and T. Pham Dinh. A dc programming approach for feature selection in support vector machines learning. *Advances in Data Analysis and Classification*, 2(3):259–278, 2008.

[42] H. A. Le Thi and M. C. Nguyen. Dca based algorithms for feature selection in multi-class support vector machine. *Ann. Oper. Res.*, 249(1):273–300, 2017.

[43] H. A. Le Thi and T. Pham Dinh. The dc (difference of convex functions) programming and dca revisited with dc models of real world nonconvex optimization problems. *Annals of operations research*, 133(1):23–46, 2005.

[44] H. A. Le Thi and T. Pham Dinh. The DC (difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems. *Annals of operations research*, 133(1-4):23–46, 2005.

[45] H. A. Le Thi, T. Pham Dinh, H. M. Le, and X. T. Vo. DC approximation approaches for sparse optimization. *European Journal of Operational Research*, 244(1):26–46, 2015.

[46] M. LeBlanc and R. Tibshirani. Combining estimates in regression and classification. *Journal of the American Statistical Association*, 91(436):1641–1650, 1996.

[47] C.-P. Lee and D. Roth. Distributed box-constrained quadratic optimization for dual linear svm. In *ICML*, pages 987–996. PMLR, 2015.

[48] C.-P. Lee and S. J. Wright. Random permutations fix a worst case for cyclic coordinate descent. *IMA Journal of Numerical Analysis*, 39(3):1246–1275, 2019.

[49] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

[50] F. Li, Y. Jin, W. Liu, B. P. S. Rawat, P. Cai, H. Yu, et al. Fine-tuning bidirectional encoder representations from transformers (bert)–based models on large-scale electronic health record notes: an empirical study. *JMIR medical informatics*, 7(3):e14830, 2019.

[51] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[52] Z. Lu and L. Xiao. On the complexity analysis of randomized block-coordinate descent methods. *Mathematical Programming*, 152(1):615–642, 2015.

[53] Y. Luan and S. Lin. Research on text classification based on cnn and lstm. In *2019 IEEE international conference on artificial intelligence and computer applications (ICAICA)*, pages 352–355. IEEE, 2019.

[54] R. Luo, L. Sun, Y. Xia, T. Qin, S. Zhang, H. Poon, and T.-Y. Liu. Biogpt: generative pre-trained transformer for biomedical text generation and mining. *Briefings in Bioinformatics*, 23(6):bbac409, 2022.

[55] H. P. H. Luu. *Advanced machine learning techniques based on DCA and applications to predictive maintenance*. PhD thesis, Université de Lorraine, 2022.

[56] H. P. H. Luu, H. M. Le, and H. A. Le Thi. Markov chain stochastic dca and applications in deep learning with pdes regularization. *Available at SSRN 4453242*, 2023.

[57] S. Makki, Z. Assaghir, Y. Taher, R. Haque, M.-S. Hacid, and H. Zeineddine. An experimental study with imbalanced classification approaches for credit card fraud detection. *IEEE Access*, 7:93010–93022, 2019.

[58] T. McIntosh and J. R. Curran. Reducing semantic drift with bagging and distributional similarity. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 396–404, 2009.

[59] X. Mi, F. Zou, and R. Zhu. Bagging and deep learning in optimal individualized treatment rules. *Biometrics*, 75(2):674–684, 2019.

[60] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[61] D. Miller. Leveraging bert for extractive text summarization on lectures. *arXiv preprint arXiv:1906.04165*, 2019.

[62] F. Mordelet and J.-P. Vert. A bagging svm to learn from positive and unlabeled examples. *Pattern Recognition Letters*, 37:201–209, 2014.

[63] F. Moretti, S. Pizzuti, S. Panzieri, and M. Annunziato. Urban traffic flow forecasting through statistical and neural network bagging ensemble hybrid modeling. *Neurocomputing*, 167:3–7, 2015.

[64] J. Mullenbach, Y. Pruksachatkun, S. Adler, J. Seale, J. Swartz, T. G. McKelvey, H. Dai, Y. Yang, and D. Sontag. Clip: a dataset for extracting action items for physicians from hospital discharge notes. *arXiv preprint arXiv:2106.02524*, 2021.

[65] Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

[66] W. W. Ng, J. Zhang, C. S. Lai, W. Pedrycz, L. L. Lai, and X. Wang. Cost-sensitive weighting and imbalance-reversed bagging for streaming imbalanced and concept drifting in electricity pricing classification. *IEEE Transactions on Industrial Informatics*, 15(3):1588–1597, 2018.

[67] J. Nutini, I. Laradji, and M. Schmidt. Let's make block coordinate descent go fast: Faster greedy rules, message-passing, active-set complexity, and superlinear convergence. *arXiv preprint arXiv:1712.08859*, 2017.

[68] R. Padaki, Z. Dai, and J. Callan. Rethinking query expansion for bert reranking. In *Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part II 42*, pages 297–304. Springer, 2020.

[69] M. P. Perrone and L. N. Cooper. When networks disagree: Ensemble methods for hybrid neural networks. In *How We Learn; How We Remember: Toward An Understanding Of Brain And Neural Systems: Selected Papers of Leon N Cooper*, pages 342–358. World Scientific, 1995.

[70] L. Petrini, F. Cagnetta, E. Vanden-Eijnden, and M. Wyart. Learning sparse features can lead to overfitting in neural networks. *Advances in Neural Information Processing Systems*, 35:9403–9416, 2022.

[71] V. T. Pham, H. A. Le Thi, H. P. H. Luu, and P. Damel. DCA-based weighted bagging: A new ensemble learning approach. In *The 15th Asian Conference on Intelligent Information and Database Systems*, Thailand, 2023. Accepted manuscript.

[72] T. Pham Dinh et al. Algorithms for solving a class of nonconvex optimization problems. methods of subgradients. In *North-Holland Mathematics Studies*, volume 129, pages 249–271. Elsevier, 1986.

[73] T. Pham Dinh and H. A. Le Thi. Convex analysis approach to DC programming: theory, algorithms and applications. *Acta mathematica vietnamica*, 22(1):289–355, 1997.

[74] T. Pham Dinh and H. A. Le Thi. Convex analysis approach to dc programming: theory, algorithms and applications. *Acta Math. Vietnam*, 22(1):289–355, 1997.

[75] T. Pham Dinh and H. A. Le Thi. A dc optimization algorithm for solving the trust-region subproblem. *SIAM Journal on Optimization*, 8(2):476–505, 1998.

[76] T. Pham Dinh and H. A. Le Thi. Recent advances in DC programming and DCA. *Transactions on computational intelligence*, 8342:1–37, 2014.

[77] D. N. Phan and H. A. Le Thi. Group variable selection via lp, 0 regularization and application to optimal scoring. *Neural Networks*, 118:220–234, 2019.

[78] A. P. Polednak and J. T. Flannery. Brain, other central nervous system, and eye cancer. *Cancer*, 75(S1):330–337, 1995.

[79] Z. Qin, K. Scheinberg, and D. Goldfarb. Efficient block-coordinate descent algorithms for the group lasso. *Mathematical Programming Computation*, 5(2):143–169, 2013.

[80] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[81] A. Rahimi and B. Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

[82] R. Rasti, M. Teshnehlab, and S. L. Phung. Breast cancer diagnosis in dce-mri using mixture ensemble of convolutional neural networks. *Pattern Recognition*, 72:381–390, 2017.

[83] L. Rokach. Ensemble-based classifiers. *Artificial intelligence review*, 33:1–39, 2010.

[84] X. Rong. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*, 2014.

[85] A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof. On-line random forests. In *2009 ieee 12th international conference on computer vision workshops, iccv workshops*, pages 1393–1400. IEEE, 2009.

[86] B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. In *COLT 2001*, pages 416–426. Springer, 2001.

[87] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[88] S. Siami-Namini, N. Tavakoli, and A. S. Namin. The performance of lstm and bilstm in forecasting time series. In *2019 IEEE International conference on big data (Big Data)*, pages 3285–3292. IEEE, 2019.

[89] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[90] M. Szummer and T. Jaakkola. Information regularization with partially labeled data. *Advances in Neural Information processing systems*, 15, 2002.

[91] S. Tammina. Transfer learning using vgg-16 with deep convolutional neural network for classifying images. *International Journal of Scientific and Research Publications (IJSRP)*, 9(10):143–150, 2019.

[92] C. A. Turner, A. D. Jacobs, C. K. Marques, J. C. Oates, D. L. Kamen, P. E. Anderson, and J. S. Obeid. Word2vec inversion and traditional text classifiers for phenotyping lupus. *BMC medical informatics and decision making*, 17(1):1–11, 2017.

[93] V. Vapnik. *The nature of statistical learning theory*. Springer science & business media, 1999.

[94] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[95] D. Veganzones and E. Séverin. An investigation of bankruptcy prediction in imbalanced datasets. *Decision Support Systems*, 112:111–124, 2018.

[96] H. Wang, J. He, X. Zhang, and S. Liu. A short text classification method based on n-gram and cnn. *Chinese Journal of Electronics*, 29(2):248–254, 2020.

[97] H. Wang, Q. Xu, and L. Zhou. Large unbalanced credit scoring using lasso-logistic regression ensemble. *PloS one*, 10(2):e0117844, 2015.

[98] K. Weiss, T. M. Khoshgoftaar, and D. Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016.

[99] S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[100] S. J. Wright. Coordinate descent algorithms. *Mathematical programming*, 151(1):3–34, 2015.

[101] H. Xiong, G. Pandey, M. Steinbach, and V. Kumar. Enhancing data analysis with noise removal. *IEEE transactions on knowledge and data engineering*, 18(3):304–319, 2006.

[102] S. Xuan, G. Liu, Z. Li, L. Zheng, S. Wang, and C. Jiang. Random forest for credit card fraud detection. In *2018 IEEE 15th international conference on networking, sensing and control (ICNSC)*, pages 1–6. IEEE, 2018.

[103] M. Zareapoor and P. Shamsolmoali. Application of credit card fraud detection: Based on bagging ensemble classifier. *Procedia computer science*, 48:679–685, 2015.

[104] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pages 818–833. Springer, 2014.

[105] J. Zhang, Y. Li, J. Tian, and T. Li. Lstm-cnn hybrid model for text classification. In *2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, pages 1675–1680. IEEE, 2018.

[106] X. Zhang, F. X. Yu, S.-F. Chang, and S. Wang. Deep transfer network: Unsupervised domain adaptation. *arXiv preprint arXiv:1503.00591*, 2015.

[107] Y. Zhang, Q. Chen, Z. Yang, H. Lin, and Z. Lu. Biowordvec, improving biomedical word embeddings with subword information and mesh. *Scientific data*, 6(1):52, 2019.

[108] L. Zhao, M. Mammadov, and J. Yearwood. From convex to nonconvex: a loss function analysis for binary classification. In *2010 IEEE international conference on data mining workshops*, pages 1281–1288. IEEE, 2010.

[109] Z. Zhao, R. Zhang, J. Cox, D. Duling, and W. Sarle. Massively parallel feature selection: an approach based on variance preservation. *Mach. Learn.*, 92(1):195–220, 2013.

[110] C. Zhou, C. Sun, Z. Liu, and F. Lau. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*, 2015.

[111] S. Zhou and W. Zhou. Unified svm algorithm based on ls-dc loss. *Machine Learning*, pages 1–28, 2021.

[112] M. Zhu, J. Xia, X. Jin, M. Yan, G. Cai, J. Yan, and G. Ning. Class weights random forest algorithm for processing class imbalanced medical data. *IEEE Access*, 6:4641–4652, 2018.

[113] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.