YOKOHAMA NATIONAL UNIVERITY
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
GRADUATE SCHOOL OF ENGINEERING

A Collaborative Machine Learning Approach for the Classification of Heterogenous and High-Dimensional Data

by

Zie Eya Ekolle

PhD Dissertation

Presented

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy (PhD) in Engineering

Date of submission: October 2023

YOKOHAMA NATIONAL UNIVERSITY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

A Collaborative Machine Learning Approach for the Classification of Heterogenous and High-Dimensional Data

PhD Dissertation Presented

by **Zie Eya Ekolle**

in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy (PhD) in Engineering

APPROVED BY:

Author: Zie Eya Ekolle

Supervisors: Prof. Hideki Ochiai and Prof. Ryuji Kohno, Yokohama National University

---

Committee Member: Prof. Tomoki Hamagami

Committee Member: Prof. Koichi Ichige

Committee Member: , Prof. Kazuhiro Otsuka

Committee Member: Prof. Masaya Nakata

Committee Member:

Department: Electrical and Computer Engineering Department, Yokohama National University

Date of submission: October 2023

Acknowledgments

First of all, I would like to express my utmost gratitude to my supervisors, Prof. Hideki Ochiai and Prof. Ryuji Kohno for guiding me through my doctoral studies and research. This also includes Prof. Kuramitsu Kimio, who guided me, alongside Prof. Ryuji Kohno, during my master course which led me to this doctoral course. Their continuous assistance, advice, and orientation were very helpful during my master and doctoral studies and research at Yokohama National University (YNU).

My special thanks go to Prof. Tomoki Hamagami, Prof. Koichi Ichige, Prof. Kazuhiro Otsuka, and Prof. Masaya Nakata. Their review and comments on my doctoral work enabled me to improve this dissertation. This also includes all the YNU course instructors and online platforms whose course materials enabled me to complete my studies and research.

I also wish to thank the members of Ochiai laboratory, Kohno laboratory, and Kuramitsu lab including the secretaries, for their sincere help in research and studies as well as in private during my master's and doctoral courses. This also includes administrative staff of the YNU student center and the Graduate School of Engineering Sciences. It was a wonderful moment with you all.

I sincerely appreciate the financial support provided by the Japanese Government (Monbukagakusho: MEXT) Scholarships during this doctoral program, and the JICA ABE scholarship during my master's degree program at YNU.

Lastly, my grateful thanks go to my parents, family members, and friends for their continuous support and tolerance throughout this period of my research.

Abstract

Classification is an important task in today's fast-growing data-generation society. The amount of data generated daily increases as more people get access to multimedia devices and networks. Coupled with the diversity of the sources, these data can be in different modalities such as text, audio, image, and video. The use of such a large amount of data almost always requires classification. Moreover, the classification task becomes challenging if such data are from heterogeneous and high-dimensional sources.

Classification involves assigning items to an appropriate predefined category. In the case of data classification, data related to some process is classified under a set of classes. An important aspect of any classification task is the classification model. The classification model defines the operations and specifications of a classification task. It is thus important to define such a model for any classification task.

Furthermore, the classification task can be automated or unautomated. In the context of the automated classification process, different types of classification models exist. This can be divided broadly into learning and non-learning models, or deterministic, stochastic, and hybrid models. In general, non-learning models are deterministic while learning models are stochastic. In this study, we focus on learning models.

A learning model carries out automated action on a target and optimizes the action during a training period. An important aspect of learning models is how they are constructed to achieve their objectives. Based on this, many approaches have been proposed in the literature related to the design and operational architecture of learning models.

In this study, we present a learning approach for the classification of heterogeneous and high-dimensional data. The approach makes use of causal and mutual actions to learn and classify data from heterogeneous and high-dimensional sources. The advantage of this approach is its high performance as compared to conventional approaches in a classification task.

Contents

Acknowledgments v
Abstract vii
Table of Contents ix
List of Figures xi
List of Tables xiii
1 Introduction 1
1.1 Background and Motivation 1
1.2 Related Work 2
1.3 Contributions of this Dissertation 3
1.4 Outline of Dissertation 3
2 Theoritical Foundation 5
2.1 Types of Machine Learning Models 6
2.1.1 Descriptive Learning Models 6
2.1.2 Predictive Learning Models 6
2.1.3 Prescriptive Learning Models 6
2.1.4 Diagnostic Learning Models 7
2.1.5 Cognitive Learning Models 7
2.2 Machine Learning Paradigms 7
2.2.1 Supervised Learning 7
2.2.2 Unsupervised Learning 16
2.2.3 Semi-supervised Learning 18
2.2.4 Self-supervised Learning 21
2.2.5 Reinforcement Learning 22
2.3 Machine Learning Approaches 24
2.3.1 Categorization of Learning Approaches 24
2.3.2 Deep Learning Approach 26
2.3.3 Ensemble Learning Approach 26
2.3.4 Federated Learning Approach 27
2.3.5 Multi-agent Reinforcement Learning Approach 28
2.3.6 Probabilsitic Learning Approach 28
2.3.7 Collaborative Learning Approach 29
2.4 Data Heterogeinety 29
2.4.1 Sources and Implications of Data Heterogeneity 30

2.4.2 Techniques to reduce Heterogeneity in Data . . . 30
2.5 The Curse of Dimensionality . . . 31
2.5.1 Origin and Implications of the Curse . . . 31
2.5.2 Techniques to avoid the Curse . . . 32
3 Modeling . . . 35
3.1 Conventional Collaborative Models . . . 35
3.1.1 Deep Belief Network . . . 35
3.2 Proposed Models . . . 37
3.2.1 General Framework . . . 38
3.2.2 Collabo: A Discriminative Collaborative Learning Approach . . . 39
3.2.3 GenCo: A Generative Collaborative Learning Approach . . . 39
3.2.4 CoPreMo: A Collaborative Predictive Model in Time Series . . . 40
3.2.5 Comparison with Conventional Models . . . 40
3.3 Performance Measures . . . 40
3.3.1 Action (Belief) Performance . . . 40
3.3.2 Learning Performance . . . 40
3.3.3 System Performance . . . 41
3.4 Statistical Signifcance . . . 41
3.5 Confidence Interval . . . 41
3.6 Model Selection . . . 41
4 Application 1: Security in Heterogeneous Medical Data using Collabo . . . 43
4.1 Background . . . 43
4.2 Model Description . . . 46
4.3 Experimental Setup . . . 48
4.4 Results and Discussion . . . 51
5 Application 2: Text Classification in NLP using GenCO . . . 59
5.1 Background . . . 59
5.2 Model Description . . . 61
5.3 Experimental Setup . . . 72
5.4 Results and Discussion . . . 73
6 Application 3: Target Tracking in ADAS/AD Radar using CoPreMo . . . 77
6.1 Background . . . 77
6.2 Model Description . . . 79
6.3 Experimental Setup . . . 82
6.4 Results and Discussions . . . 83
7 Conclusion . . . 85
Bibliography . . . 87
Appendices
A Publications . . . 91

List of Figures

2.1 Supervised machine learning paradigm. 8
2.2 Unsupervised machine learning process. 16
2.3 Reinforcement learning process. 22

3.1 DBM architecture with RBM and Logistic regression layers. 36

4.1 Medical IoT network. 44
4.2 DDoS attack operations in an IoT network. 45
4.3 DDoS attack detection and prevention operations in an IoT network with the proposed solution. 46
4.4 Cost-based learning curve of our model on DDoS attack detection compared to other models using ECU-IoHT training dataset. 51
4.5 Accuracy-based learning curve of our model on DDoS attack detection compared to other models using ECU-IoHT training dataset. 52
4.6 ROC curve of our model on DDoS attack detection compared to other models using ECU-IoHT test dataset. 52
4.7 Cost-based learning curve of our model on DDoS attack detection compared to other models using ICU training dataset. 53
4.8 Accuracy-based learning curve of our model on DDoS attack detection compared to other models using ICU training dataset. 54
4.9 ROC curve of our model on DDoS attack detection compared to other models using ICU test dataset. 54
4.10 Cost-based learning curve of our model on DDoS attack detection compared to other models using ToN_IoT training dataset. 55
4.11 Accuracy-based earning curve of our model on DDoS attack detection compared to other models using ToN_IoT training dataset. 55
4.12 ROC curve of our model on DDoS attack detection compared to other models using ToN_IoT test dataset. 56

5.1 Text classification using Naive Bayes. 62
5.2 Text classification using GenCo. 68

5.3 Results with the Twitter US Airline dataset. (**a**) Confusion matrix. (**b**) Mutuality matrix for the 10 feature segments. (**c**) Combined mutuality of each feature in the segment with the highest mutuality. 73
5.4 Results with the Conference Paper dataset. (**a**) Confusion matrix. (**b**) Mutuality matrix for the 10 feature segments. (**c**) Combined mutuality of each feature in the segment with the lowest mutuality. 75
5.5 Results with SMS Spam dataset. (**a**) Confusion matrix. (**b**) Mutuality matrix for the 5 feature segments. (**c**) Combined mutuality of each feature in the segment with the lowest mutuality. 75

6.1 ADAS vehicle with sensors location based on traffic scenario. 78
6.2 General architecture for a radar system and control unit. 79
6.3 Tracking of radar properties using CoPreMo on a 1D input data 80
6.4 Simulation of an ADAS/AD vehicle collision scenario with corner radar detection and tracking. 83

List of Tables

4.1 List of attacks and input fields used in the ECU-IoHT dataset 48
4.2 List of classes and input fields used in the ICU dataset 49
4.3 List of attacks and input fields used in ToN_IoT (Network) dataset 49
4.4 Partition of attack instances into training and test instances 50
4.5 Simulation parameters of our model and conventional models 50
4.6 Predictive performance and comparison of the models 56
4.7 Computational performance and uncertainty of the models 57
4.8 Comparison of our model with models from other research works 58

5.1 Bag-of-Words (i.e., 1-gram word) vectorization of a news corpus. 64
5.2 Binary Bag-of-Words (i.e., 1-gram word) vectorization of a news corpus. . . . 65
5.3 Statistics of the datasets. 72

6.1 Performance and numerical comparison. 84

Chapter 1
Introduction

1.1 Background and Motivation

Due to the rise in globalization, human interactions have increased over the past years [1]. Coupled with the increase in electronic communication, the amount of electronic data generated has surged. This electronic data can be in different modalities such as sound, image, video, or text. In almost all cases, for any operation carried on the data, a classification of the data is required to effectively understand the data content. Such classification becomes challenging if the data source is heterogeneous and high-dimensional [2].

A data classification task involves assigning a data set to an appropriate predefined category. This can be the classification of text in a corpus, detection of cyberattacks, or making control decisions in autonomous driving. In general, classification is a type of predictive action where the values of the predefined category are discrete.

To facilitate the classification task, many tools are being used such as automated learning tools, that is, machine learning tools. One primary objective of the application of machine learning approaches to classification is to train a classifier on a task to attain a high level of generalization on the task. Due to the proliferation and increased performance of machine learning algorithms, most classification tasks can be done by these algorithms.

However, some challenges still exist with machine learning applications for classification, especially when the dimensions of the source features or the target features are too large. The increase in the dimension of source features leads to a phenomenon called "the curse of dimensionality" [3, 4]. This is a bottleneck situation for machine learning algorithms. The case for many target features (multi-targets) is also a challenge in current machine learning models, especially when the dimension of the target features increases.

Apart from dimensionality constraints, machine learning algorithms have limitations related to the type of features used at the source or target. Learning from unrelated sources or learning unrelated targets, are challenging tasks for current machine learning algorithms. Such datasets where their features are different in form, for example where their features exist in different modalities, are considered heterogenous datasets [5].

Heterogenous datasets have become increasingly popular in recent years due to diver-

sity in multimedia and content creation. So, developing tools that can effectively take into account such diversity during classification is vital in modern machine learning.

The main idea of learning algorithms is to train a learning model with source data in order to make predictions on target data. The training process is actually a feedback optimization and/or update process, and many training algorithms have been developed such as gradient descent, genetic algorithms, contrastive divergence, and so on, to train different types of learning agents. In general, machine learning algorithms can be broadly divided into supervised, unsupervised, and reinforcement learning algorithms. Actually, most of the machine learning algorithms are designed and used for low-dimensional and less heterogeneous datasets. This study presents a novel model for learning from heterogeneous and high-dimensional datasets using a collaborative learning approach.

Literally, collaboration is the process where two or more entities work together, with the same or different purpose, to complete a common action (e.g., mission or task) or achieve a common target (e.g., vision, goal, objective, outcome, or output). Thus, agents can collaborate on a target or on an action, but since target and action are linked by a purpose, the collaboration on an action entails a collaboration on a target, and vice-versa. In this way, a collaborative action on a target may be divided into multiple partial actions defined by the nature of the target perceived by the agent. In this study, we use classification as the partial action on the target to achieve a collaborative prediction on the target. However, partial regression and classification can also be used collectively.

1.2 Related Work

Many research works have been done on collaborative learning models and their applications. As presented in the previous section, collaborative models are models that constitute a collection of computational units working to achieve a common objective. This can range from conventional neural networks to bayesian and markov networks. We present in this section, selected collaborative models and thier applications.

Yu et al. [6] proposed a propose a weakly-supervised method that addresses the lack of training data in neural text classification by using a dual-module for pseudo-text generation and sel training. They performed three experiments with the model and the results out performed those of the baseline models.

Li et al. [7] proposed a text classification model based on the Bidirectional Encoder Representations from Transformers (BERT) model and feature fusion. A comparison with the state-of-the-art model showed that the accuracy of the proposed model outperformed those of state-of-the-art models. The model can improve the accuracy of tag prediction for text data with sequence features and obvious local features.

Mona et al. [8] proposed a solution based on mutual information and random forest feature importance. They used these methods for feature selection to reduce the DDoS

misclassification of different machine learning models. Their results showed that such feature selection methods lead to a higher detection accuracy of DDoS using machine learning models. Maslan et al. [9] proposed a similar feature selection technique that uses a regression model called max-dependency.

Neto et al. [10] proposed a collaborative DDoS detection solution for a general IoT system, including e-health, by using federated learning. They used different deep-learning instances for each pool of data sources to generate the local parameters. To obtain a global result for the overall system, all local parameters were combined and optimized using federated learning techniques.

Du et al. [11] proposed an attention-based recurrent neural network for text classification. The network was trained on two news classification datasets published by NLPCC2014 and Reuters, respectively. The classification results showed that the model outperformed baseline models by achieving F-values of 88.5% and 51.8% on the two datasets

Awan et al. [12] proposed a real-time DDoS detection using random forest (RF) and multi-layer perceptron (MLP) machine learning models. Their solution was implemented with and without big data, but the big-data solution outperformed the non-big-data solution.

Akhter et al. [13] proposed a document classification model for the Urdu language using a single-layer multisize filters convolutional neural network (SMFCNN). They compared this model with sixteen machine learning baseline models on three imbalanced datasets. Their method achieved a higher accuracy than the selected baseline models, with accuracy values of 95.4%, 91.8%, and 93.3% on medium, large, and small size datasets, respectively.

1.3 Contributions of this Dissertation

The key contribution of this study is the proposition of a collaborative learning model based on causal and mutual value and their exchanges between learning models (agents) with different source features. Due to such value exchange, the different models can be trained together collaboratively on a common target, and their resultant (collaborative) action on the target is the accumulated action of their individual actions on the target.

1.4 Outline of Dissertation

The rest of this dissertation is organized in the following way.

- Chapter 2 entitled, "Theoritical Foundation", focuses on the theoretical foundation of machine learning. This include machine learning types, paradigm, and approaches. The different type of belief actions, error, and learning techniques for each learning

approach was presented. Also the sources, implications and solutions to data heterogeneity and cursed of dimensionality were presented.

- Chapter 3 entitled, "Modeling", focuses on the modeling approach of solutions related to the collaborative machine learning. This includes both conventional and proposed collaborative learning approaches. The general framework of the proposed model was presented from which three proposed models, Collabo, Genco, and CoPreMo were introduced. The development and applications of these three models are presented in the next chapters.

- Chapter 4 entitled, "Security in Heterogenous Medical Data Using Collabo", focuses on application of one of the Collabo model to the security of heterogenous medical data in an internet of medical things network. This involve the use of a discriminative approach to deploy the collaborative framework in medical IoT security environment. The performance of the model is evaluated and compared with other models which have been developed to address same problem.

- Chapter 5 entitled, "Text Classification in NLP using GenCo", focuses on the application of the GenCo model to the classification of text in natural language processing. This include the use of a generative approach to deploy the collaborative framework in natural language processing. The performance of the model is compared with other models which have been developed to address same problem.

- Chapter 6 entitle, "Target Tracking in ADAS/AD Radar using CoPreMo", focuses on the application of the CoPreMo model to the tracking of target in ADAS/AD Radar. This include the use of a time series approach to deploy the collaborative framework in radar tracking systems for optimal ADAS/AD performance.

- Chapter 7 is the conclusion of this dissertation.

Chapter 2
Theoritical Foundation

Machine learning is a discipline that involves the building of models (or machine-based entities) that can be trained to achieve a target by carrying out an action. Different logical frameworks have been developed over the years that are used for building machine learning models. These logical frameworks include propositional logic, probabilistic logic, statistical logic, and more recently system logic. Each of these frameworks has its advantages and limitations. The key issue is how they execute the learning process and define the target to achieve and the action to execute/learn. The target can be a problem to solve, a project to complete, an objective to attain, etc. The action can be a task to execute, an operation to perform, a process to roll out, etc.

Machine learning models built using these logical frameworks can be categorized, with respect to the nature of the target they are designed to achieve, into the following categories: supervised, unsupervised, semi-supervised, self-supervised, and reinforcement learning. This categorization defines the different machine learning paradigms. Also, another categorization of the models can be done with respect to the type of action they are designed to execute/learn, such as descriptive, diagnostic, predictive (i.e., classification or regression), prescriptive, and cognitive actions. This defines the different machine learning types. Furthermore, a categorization of the models can be done with respect to the learning techniques used by the models, such as deep learning, ensemble learning, federated learning, transfer learning, multi-task learning, and multi-target learning. This categorization defines the different machine-learning approaches.

In this dissertation, a novel type of machine learning approach called collaborative learning is presented. It is based on a hybrid logic of propositional, probabilistic, statistical, and system logic. The theoretical framework of this hybrid logic is based on the mathematical theory for intelligent agents presented in [14]. Also, the dissertation focuses on applying this approach to predictive action, precisely classification, in a supervised learning paradigm. The theoretical foundation of machine learning with regard to the collaborative approach and its applications is presented in this chapter. This theoretical foundation includes machine learning types, machine learning paradigms, machine

learning approaches, performance evaluation of learning models, data heterogeneity, and the curse of dimensionality.

2.1 Types of Machine Learning Models

The different types of machine learning models in this dissertation are based on the type of action the agent can execute on the target. These actions include description, prescription, prediction (classification and regression), diagnostic, and cognition. Each of these actions represents a different type of machine learning model. Actually, predictive models are predominant in current AI research literature and industry.

However, this dissertation focuses on Cognitive modeling, which in reality is the last dimension of artificial intelligence and machine learning design, because it is required to encompass all the other actions. Simply put, a cognitive agent is an agent with human-like properties, that is a humanoid. It is thus required to take different types of actions, ranging from prescription, description, prediction, and diagnostic, just as a human agent.

2.1.1 Descriptive Learning Models

This is a learning approach that aims to optimize the descriptive actions of an artificial intelligence and machine learning agent. Descriptive actions or analytics is the process of using current and past data to identify trends and relationships between variables within the data. It is generally a noncausal action. One common descriptive method is clustering. Others include the use of different statistical measures such as the measure of central tendency, dispersion, and position.

2.1.2 Predictive Learning Models

This is a learning approach that aims to optimize the predictive actions of an artificial intelligence and machine learning agent. Predictive actions or analytics is an action that uses current and historical data to forecast or predict the outcome of a given target (e.g., problem, activity, behavior, trends, etc). It is generally a causal action. This is the most popular type of machine learning in recent years and is mainly divided into regression and classification actions. Examples thus include logistic regression, causal deep neural networks, bayesian inference, etc.

2.1.3 Prescriptive Learning Models

This is a learning approach that aims to optimize the prescriptive actions of an artificial intelligence and machine learning agent. Prescriptive actions or analytics is the process of using data to recommend an optimal course of action. Prescriptive models differ from

predictive models in that apart from predicting the outcome, they also suggest actions to benefit from the predictions and show the implications of each decision option. Hence, prescriptive models use both descriptive and predictive actions. A common descriptive method is a recommendation system.

2.1.4 Diagnostic Learning Models

This is a learning approach that aims to optimize the diagnostic actions of an artificial intelligence and machine learning agent. Diagnostic actions use past data to understand an outcome and provide answers to why an outcome occurs. Thus, it focuses on both the cause and effect of a target. It takes into account descriptive, predictive, and prescriptive actions. It uses techniques such as drill-down, data discovery, data mining, and correlations. Developing a true diagnostic model is still an ongoing process.

2.1.5 Cognitive Learning Models

This is a learning approach that aims to optimize the cognitive actions of an artificial intelligence and machine learning agent. Cognitive action or analytics involve the simulation of the human thought process to learn from the data and extract the hidden patterns from data to describe, predict, prescribe, and diagnose a target. It is the most complex action in machine learning, and the development of a true cognitive model is still a research challenge.

2.2 Machine Learning Paradigms

Machine learning paradigms can be broadly categorized into supervised, unsupervised, semi-supervised, self-supervised, and reinforcement learning.

2.2.1 Supervised Learning

Supervised learning [15, 16] is a common learning paradigm. In supervised learning, the model learns from a set of input-output pairs of data, which are called labeled datasets. The objective of supervised learning is usually to train a predictive model using these labeled datasets.

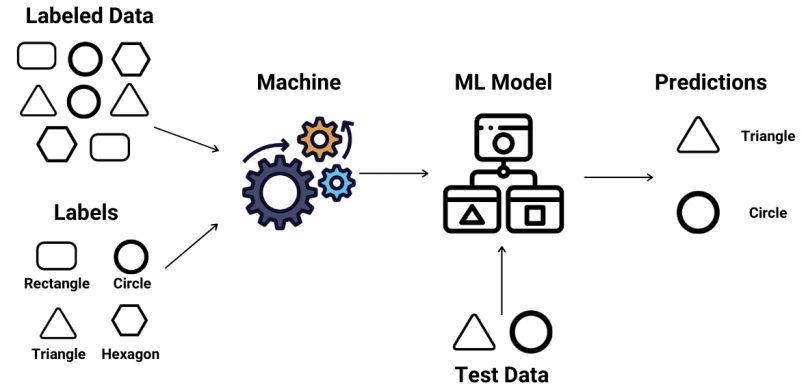


Figure 2.1: Supervised machine learning paradigm.

In general, a supervised learning task can be represented as a function that is trained to map input data to output data using provided input-output data pairs. So, a supervised learning model definitely needs the input-output data pairs during training. Depending on the nature of the training function, different supervised learning models have been developed such as logistic regression, deep neural network, naive Bayes, and so on. Supervised learning models can be generally defined as described below.

Given a dataset $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, where each x_i is a d-dimensional input (or feature vector), and y_i is the corresponding output called label define under the space S. It is assumed that these data points are drawn from some unknown distribution P, that is, $(x_i, y_i) \sim P$. Then, the objective of supervised machine learning is to find a function $f : \mathbb{R}^d \rightarrow S$, such that for every input-output value pair,

$$f(X) = y \tag{2.1}$$

where $X = (x_1, x_2, ..x_n)$ is a vector of input values, x_i is the feature vector of the ith input example, $\mathbb{R}^d$ is the d-dimensional input feature space, y_i is the feature vector of the ith output example, S is the space of all possible labels (i.e., label space), and n is the size of the dataset.

Different causal functions are used to define the function f. Some examples of the causal functions used in supervised learning include the linear function, logistic (or sigmoid) function, sign function, rectangular function, softmax function, softplus function, relu function, elu function, tanh function, step function, conditional probability function, and so on. Some of these functions can be interconnected to form a network of computational causal models such as the neural network, Bayesian(or belief) network, and Markov network. These unit causal functions are also called activation functions in neural network models and kernel functions in support vector machines (SVM). In this dissertation and in accordance with [14], they are considered as action functions and their output value is action value. Below is a list of some unit causal functions in both statistical and probabilistic causal networks.

A) Action Functions

i) Linear function
ii) Sigmoid function
iii) Softmax function
iv) Softplus function
v) Relu function
vi) Elu function
vii) Tanh function
viii) Step function
ix) Rectangular function
x) Conditional probablity function

Using these causal functions, to achieve the objective of relating the input X and output y, a learning process is used. Depending on the type of supervised learning model, an optimization and/or an update approach can be used for training. In most optimization approaches such as gradient descent for logistic regression and deep neural network models, the optimization of a loss function is required. The value of the loss function tells us how good the function f is, given the data D which it is required to reproduce. Many loss functions have been developed over the years for different supervised learning problems and models such as the cross-entropy (CE) loss, Kullback-Leibler divergence (KLD) loss, mean square error loss (MSE), mean absolute error (MAE) loss, Huber loss, and hinge loss.

However, some models do not use differential optimization but also make use of values to evaluate and guide their learning process. In this dissertation, all values used to learn a model are called learning values or cost values, which the agent may seek to reduce (if considered as a loss) or increase (if considered as a gain) during learning. In supervised learning, they are mostly expressed as an error function of the action value on the target with respect to the actual value of the target. Below is a description of some of the learning values used in supervised learning.

B) Learning Values

i) Cross entropy (CE) loss (or negative log loss): This is a common loss function in machine learning, especially for classification tasks, such as binary output values. It is an information theoretical value, deployed in machine learning as a loss function to capture the generalization of a supervised learning process over a given dataset. It uses a probabilistic approach to measure the loss value of a learning model by quantifying the logarithmic expectation of the reciprocal predicted probability with respect to the actual probability of each output instance over the dimension (i.e., cardinality) of the discrete output space as shown in Eq(2.2). To capture the CE loss value of the model over the entire instances of the training dataset, the average cross entropy loss over all the instances of the training

dataset, as shown in Eq(2.3), is commonly used.

Given that the function in Eq(2.23) is defined as a probabilistic causal function using conditional probability, $f(X) = P(y|X) = p$, then,

$$\text{Instance Cross Entropy (ICE)} = \sum_{i=1}^{m} p_i \log \frac{1}{\hat{p}_i} = -\sum_{i=1}^{m} p_i \log \hat{p}_i \tag{2.2}$$

$$\text{Batch Cross Entropy (BCE)} = \frac{1}{n} \sum_{j=1}^{n} \left(\sum_{i=1}^{m} p_{ij} \log \frac{1}{\hat{p}_{ij}} \right) = -\frac{1}{n} \sum_{j=1}^{n} \left(\sum_{i=1}^{m} p_{ij} \log \hat{p}_{ij} \right) \tag{2.3}$$

where m is the number of classes, where n is the number of output instances in the dataset, p is the true probability of the label, $\hat{p}$ is the predicted probability of the label, and $1/\hat{p}$ is the reciprocal probality of the label according to p.

However, with respect to the dimension of the output space and computational complexity of calculating (e.g., differentiating) the cross-entropy loss, different variants of the cross-entropy loss are mostly used, such as the binary cross-entropy loss and the categorical cross-entropy loss. The binary cross-entropy loss, shown in Eq(2.4), is used when the output space is binary, i.e., having a 2-dimensional space with mutually exclusive values. On the other hand, the categorical cross-entropy, shown in Eq(2.5), is used for any dimension of the output space, with or without mutual exclusiveness between its values, such as in multi-class and multi-label classifications.

$$\text{Batch Binary Cross Entropy (BBCE)} = -\frac{1}{n} \sum_{j=1}^{n} \left(p_j \log \hat{p}_j + (1 - p_j) \log \hat{p}_i \right) \tag{2.4}$$

$$\text{Batch Categorical Cross Entropy (BCCE)} = -\frac{1}{n} \sum_{j=1}^{n} \left(\sum_{i=1}^{m} p_{ij} \log \hat{p}_{ij} \right) \tag{2.5}$$

where m is the number of classes, where n is the number of output instances in the dataset, p is the true probability of the label label, $\hat{p}$ is the predicted probability of the label, and $1/\hat{p}$ is the reciprocal probality of the label according to p.

ii) Kullback-Leibler divergence (KLD) loss (or relative entropy loss): It is an information-theoretic measure that is based on probabilistic logic and is commonly used to compare two probability distributions. This divergence property enables it to be used in many machine learning operations such as the loss value of a machine learning model, especially for classification tasks. As a loss value, it quantifies the logarithmic expectation of the actual and predicted probability ratio with respect to the actual probability of each output instance over the dimension (i.e., cardinality) of the discrete output space as shown in Eq(2.6). To capture the KLD loss value of the model over the entire instances of the training dataset, the average cross entropy loss over all the instances of the training dataset, as

shown in Eq(2.7), is commonly used.

Given that the function in Eq(2.23) is defined as a probabilistic causal function using conditional probability, $f(X) = P(y|X) = p$, then,

$$\text{Instance KLD (IKLD)} = \sum_{i=1}^{m} p_i \log \frac{p_i}{\hat{p}_i} = -\sum_{i=1}^{m} p_i \log \frac{\hat{p}_i}{p_i} \tag{2.6}$$

$$\text{Batch KLD (BKLD)} = \frac{1}{n}\sum_{j=1}^{n}\left(\sum_{i=1}^{m} p_{ij} \log \frac{p_{ij}}{\hat{p}_{ij}}\right) = -\frac{1}{n}\sum_{j=1}^{n}\left(\sum_{i=1}^{m} p_{ij} \log \frac{\hat{p}_{ij}}{p_{ij}}\right) \tag{2.7}$$

where m is the number of classes, where n is the number of output instances in the dataset, p is the true probability of the label label, $\hat{p}$ is the predicted probability of the label, $p/\hat{p}$ is the relative probality of the label according to p compared to $\hat{p}$, and $\hat{p}/p$ is the relative probality of the label according to $\hat{p}$ compared to p.

iii) Mean square error (MSE) loss: It is a regression-based technique to evaluate the loss of a learning model. It measures the average squared difference between the actual value and the estimated value.

Given that the function in Eq(2.23) is defined as a statistical causal function using regression model, $f(X) = g(X, \theta) = a + \theta X + e = y$, then,

$$\text{Mean square error (MSE)} = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{2.8}$$

where n is the number of output instances, y is the true label, $\hat{y}$ is the predicted label, θ is the vector of regression weights, a is the intercept, and e is the regression error.

iv) Mean absolute error (MAE): It is also a regression-based technique to evaluate the loss of a learning model. It measures the average absolute difference between the actual value and the estimated value.

Given that the function in Eq(2.23) is defined as a statistical causal function using regression model, $f(X) = g(X, \theta) = a + \theta X + e = y$, then

$$\text{Mean absolute error (MAE)} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{2.9}$$

where n is the number of output instances, y is the true label, $\hat{y}$ is the predicted label, θ is the vector of regression weights, a is the intercept, and e is the regression error.

v) Huber loss: This is also a statistical regression loss function used mostly for robust regression rather than MSE and MAE because it contains both MSE and MAE properties, and is less sensitive to outliers in data than MSE and MAE. It defines the convolution of an absolute value function with a rectangular function, through scaling and translation.

Unlike MSE and MAE losses that are mathematically defined by default over a batch of instances, Huber loss has separate mathematical definitions for instance and batch measures as shown in Eq(2.10) and (2.11), respectively.

Given that the function in Eq(2.23) is defined as a statistical causal function using regression model, $f(X) = g(X, \theta) = a + \theta X + e = y$, then,

$$\text{Instance Huber loss} = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & \text{for } |y - \hat{y}| \leq \delta \\ \delta.(|y - \hat{y}| - \frac{1}{2}\delta), & \text{otherwise} \end{cases} \tag{2.10}$$

$$\text{Batch Huber loss} = \begin{cases} \frac{1}{n}\sum_{i=1}^{n} \frac{1}{2}(y_i - \hat{y}_i)^2 & \text{for } |y_i - \hat{y}_i| \leq \delta \\ \frac{1}{n}\sum_{i=1}^{n} \delta_i.(|y_i - \hat{y}_i| - \frac{1}{2}\delta_i), & \text{otherwise} \end{cases} \tag{2.11}$$

where δ is the point where the Huber loss function changes from a quadratic to linear, n is the number of output instances, y is the true label, $\hat{y}$ is the predicted label, θ is the vector of regression weights, a is the intercept, and e is the regression error.

vi) Hinge loss: This is a statistical loss function mostly used for classification in situations where we want to separate a group of data points from another group. This is a typical technique for learning models such as Support vector machines (SVM) that use a datapoint abstraction approach for learning.

Given that the function in Eq(2.23) is defined as a statistical causal function using regression model, $f(X) = g(X, \theta) = a + \theta X + e = y$, assuming $y \in \{-1, 1\}$, then

$$\text{Instance Hinge loss} = \max(0, 1 - y\hat{y}) \tag{2.12}$$

$$\text{Batch Hinge loss} = \sum_{i=1}^{n} \max(0, 1 - y_i\hat{y}_i) \tag{2.13}$$

where n is the number of output instances, y is the true label, $\hat{y}$ is the predicted label, θ is the vector of regression weights, a is the intercept, and e is the regression error.

If $|\hat{y}| \geq 1$ and have same sign as y, then $y\hat{y} > 1$ and the loss is zero. If $0 \leq |\hat{y}| < 1$ and the prediction is correct, then the hinge loss still attributes some loss value to the model to penalize the model for making less certain predictions.

One issue with the hinge loss is that it is not differentiable and hence cannot be learned using differential optimization techniques such as gradient descent. However, because it is a convex function, we can definitely find its global optimum. Also, Hinge loss is robust for multiclass classification.

vii) 0-1 loss: It counts the number of mistakes the predictive (or causal) function makes on the training dataset during learning. As defined in Eq(2.14), for every single instance of the training dataset, it attributes a loss of 1 if it is mispredicted, and 0 otherwise. The normalized 0-1 loss returns the fraction of misclassified training instances, referred to in

most cases as the training error.

$$\text{Batch 0-1 loss} = \frac{1}{n}\sum_{i=1}^{n}\delta_{\hat{y}_i \neq y_i}, \text{ where } \delta_{\hat{y}_i \neq y_i} = \begin{cases} 1, & \text{if } \hat{y}_i \neq y_i \\ 0, & \text{if } \hat{y}_i = y_i \end{cases} \tag{2.14}$$

where n is the number of output instances, y is the true label, and $\hat{y}$ is the predicted label.

This loss is non-differentiable and suitable for non-differential optimization learning approaches such as Bayesian models for supervised learning and K-means models for unsupervised learning. However, it is used for both classification (discrete output) and regression (continuous output) problems. It is also used as a prediction performance measure for multi-label classification.

viii) Hamming loss: It defines the fraction of the wrong predicted labels to the total number of labels as shown in Eq(2.15). It is similar to the 0-1 loss in that both perform counts operations, but unlike the 0-1 loss that can be used for both classification and regression, the hamming loss is mostly used to evaluate the learning and predictive performance of classification models such as multilabel classification models.

$$\text{Batch Hamming loss} = \frac{1}{kn}\sum_{i=1}^{k}\sum_{j=1}^{n}\text{xor}(\hat{y}_{ij}, y_{ij}), \text{ where } \text{xor}(\hat{y}_{ij}, y_{ij}) = \begin{cases} 1, & \text{if } \hat{y}_i \neq y_i \\ 0, & \text{if } \hat{y}_i = y_i \end{cases} \tag{2.15}$$

where k is the number of labels, n is the number of output instances of a label, y is the true label, $\hat{y}$ is the predicted label, and xor(.) indicate exclusive OR operator.

ix) Gini impurity (or Gini's diversity index): It is a learning value used in decision tree classification models to estimate the optimal decision (or split) node from a root node. It measures the probability of misclassifying an observation instance if it were labeled randomly and independently according to the distribution of labels in the dataset. It reaches a minimum impurity of zero when all instances in the node belong to a single class (i.e., homogenous), and has the highest impurity of 1 when all instances in a node belong to a different class (i.e., heterogenous). The Gini Impurity at each decision node (or feature) a is calculated using the following formula,

$$\text{Gini impurity} = 1 - \sum_{i=1}^{m}(p(c_i|a = v))^2 \tag{2.16}$$

where m is the number of classes, c is a class, and $p(c_i|a = v)$ is the probability at a given decision node a with value v, to choose an observation instance belonging to class c_i.

This loss can be considered as an information-theoretic measure corresponding to the Tsallis entropy, which is a generalization of the standard Boltzmann–Gibbs entropy and the Shannon entropy. Furthermore, the Gini impurity is actually used to quantify both the

learning and prediction performances of a classification model. More so, it can be used to quantify the degree of heterogeneity and homogeneity in a given dataset.

x) entropy (or expected information): This is an information-theoretic measure based on probability and is commonly used to measure the amount of uncertainty or expected information content in an information system. It is used as a learning value in many learning models such as decision tree models. In a decision tree, unlike the Gini impurity which uses a heterogeneity-based approach for decision-making, the entropy value measures the homogeneity of a node. It is defined as follows,

$$\text{Entropy} = -\sum_{i=1}^{m} p(c_i|t)\log p(c_i|t) \tag{2.17}$$

where m is the number of class c, and $p(c_i|t)$ is the probability of a class c_i at a node t.

It has a minimum value (i.e., more information) when all instances belong to the same class and a maximum value (i.e., least information) when instances are equally distributed.

xi) information gain (or mutual information): Similar to the Gini impurity, this learning value is also mostly used for decision tree models. Furthermore, it is an information-theoretic measure based on probability and thus can be related to CE, KLD, Gini impurity, and definitely entropy. The expected (or average) information gained over the set of classes is equivalent to the mutual information. The information gain of a root node t is obtained from an observation of a decision node a having a class c at value $a = v$.

$$\begin{aligned} &\text{Information gain} = H(t) - H(t|a) \\ &\text{Given that, } H(t) = -\sum_{i=1}^{m} p(c_i)\log_2 p(c_i) \text{ and } H(t|a) = -\sum_{i=1}^{m} P(c_i|a)\log_2 P(c_i|a) \end{aligned} \tag{2.18}$$

where t is the root node, a is a decision node, m is the number of classes, $p(c_i)$ is the probability (or fraction) of a class c_i present in the parent node that results from a split in the tree, $P(c_i|a)$ is the probability (or fraction) of a class c_i present in the child node that results from a split in the tree, $H(t)$ entropy of parent node, and $H(t|a)$ is the weighted or average entropy at children nodes.

The expected information gain can also be estimated over a vector of decision nodes $A = (a_1, a_2, ..a_j)$ at each root node t, in which case,

$$\begin{aligned} &\text{Information gain over A} = H(t) - H(t|A) \\ &\text{Given that, } H(t) = -\sum_{i=1}^{m} p(c_i)\log_2 p(c_i) \text{ and } H(t|A) = \sum_{j=1}^{n} p(a_j)H(t|a_j) \end{aligned} \tag{2.19}$$

In general, the expected information gain approach consists of reducing information

entropy $H(t)$ of the root node t from a prior state to a state that has more information, i.e., more entropy (or uncertainty). Once calculated at each node, the information gain is used to decide which input features to use in splitting at each decision node in building the decision tree, and the split with the highest information gain is considered first. The process will continue until all children nodes have consistent data.

After defining the learning value(s) for any given supervised learning model, the next step is to define the training process of the learning model. The training process of a supervised learning model can generally be defined as an optimization problem as follows,

$$\begin{aligned} &\underset{X,\, y}{\text{optimize}}\ \mathcal{H}(X, y) \\ &\text{subject to } \mathcal{K} \end{aligned} \tag{2.20}$$

where $\mathcal{H}$ is the optimization function (i.e., objective function), and $\mathcal{K}$ is the optimization parameter (i.e., constraint).

For update-based learning, $\mathcal{H}(X, y)$ is the inference function, that is, $\mathcal{H}(X, y) = f(X, y)$, and $\mathcal{K}$ is the size of the dataset. In Bayesian updating, increasing the size of the dataset maximized the value of $f(X, y)$, making it closer to the true value. On the other hand, for optimization-based learning, $\mathcal{H}(X, y)$ is the learning function and $\mathcal{K}$ is the prediction function. In this case, $\mathcal{H}$ is mostly considered as a function of $\mathcal{K}$, and depending on the type of optimization algorithm and loss value, $\mathcal{H}$ can be maximized or minimized to make $\mathcal{K}$ closer to the true value. For example, giving a gradient descent optimization and a cross-entropy loss value, $\mathcal{H}$ is minimized to make $\mathcal{K}$ closer to the true value.

The optimization technique can be derivative-based such as gradient descent, momentum, adagrad, adamax, adadelta, adam, nesterov, RMSprop, conjugate gradient, quarsi-newton, etc, or derivative-free such as bayesian updates, bayesian optimization, particle swarm optimization, random search, genetic algorithm, differential evolution, simulated annealing, etc. Derivative-based optimization always requires a differentiable learning value but a derivative-free optimization does not require such conditions. In machine learning, these optimization techniques are usually nonlinear optimization/programming techniques, which can therefore be grouped into local search methods such as gradient-based (i.e., derivative-based) methods, and global search methods such as metaheuristics (i.e., derivative-free) methods. Below is a list of some of these techniques.

C) Learning Techniques

i) Gradient descent:

ii) Gradient ascent:

iii) Momentum

iv) AdaGrad

v) Adam
vi) RMSprop
vii) Conjugate gradient
viii) Bayesian update
ix) Bayesian optimization
x) AdaBoost

These algorithms are used mainly for learning a single computational causal function. Apart from such learning algorithms for single computational units, learning techniques for an entire network of computational units also exist for different types of networks such as the backpropagation learning technique for deep neural networks (feedforward, recurrent, recursive, or convolution), belief propagation update technique for graphical models (Bayesian or Markov networks), genetic algorithm, particle swarm optimization, simulated annealing, and differential evolution for evolutionary intelligence networks. It is in this learning framework that we introduce a collaborative learning approach presented in this dissertation. More on these will be explained in Chapter 3, where I discussed the conventional and proposed collaborative models and their learning approaches.

2.2.2 Unsupervised Learning

In unsupervised learning [15, 17], the datasets are not labeled. In fact, there are neither inputs nor outputs, but a set of samples with features. Unsupervised learning can be used for many tasks such as clustering, which separates data examples into groups called clusters and is mostly used as a data exploration tool.

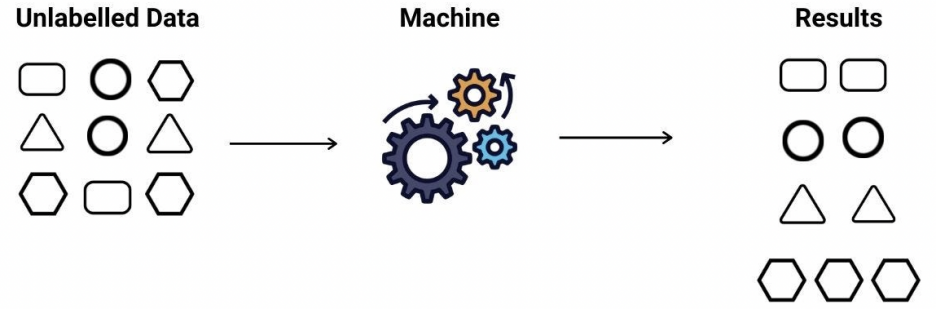


Figure 2.2: Unsupervised machine learning process.

In general, an unsupervised learning task can be represented as a function that is trained to map real (visible) input data to an abstract (hidden) output without any real input-output pairs definition. So, an unsupervised learning model does not need the input-output pairs of data during training. Furthermore, unlike supervised learning models whose learning values are explicitly defined, the learning values used by most unsupervised learning models are not explicitly defined. Learning values such as the 0-1 loss val-

ues are used in some unsupervised learning models such as the K-means learning model. Unsupervised learning models can generally be described as explained below.

Given a dataset $\tilde{D} = \{x_1, \ldots, x_n\}$, where each x_i is a d-dimensional input or feature vector. It is assumed that these data points are drawn from some unknown distribution $\tilde{P}$, that is, $(x_i) \sim \tilde{P}$. Then, the objective of unsupervised machine learning is to find a function $\tilde{f} : \mathbb{R}^d \rightarrow \tilde{S}$, such that for every input vector,

$$\tilde{f}(X) = \tilde{y} \tag{2.21}$$

where X is a vector of input values, x_i is the feature vector of the ith input example, $\mathbb{R}^d$ is the d-dimensional input feature space, $\tilde{y}$ is an abstract output feature, $\tilde{S}$ is the space of the abstract output feature i.e., abstract space, and n is the size of the dataset.

Similar to supervised learning, the function f in unsupervised learning can be defined using different functions. However, unlike the functions used in supervised learning, the functions used in unsupervised learning are mostly non-causal in the real domain and causal in the abstract domain. This imply that, they defined only abstract causalal relationship and not real causal relationship. Such functions can be defined differently using different logic and tool. For example, in the context of probability logic, a non causal function can be defined using the marginal distribution, the joint distribution, and so on. Below are some non causal action functions used for unsupervised learning.

A) Action Functions

i) Marginal probability distribution functions
ii) Joint probability distribution functions
iii) Correlation function
iv) Covariance function
v) Gibbs measure

Training an unsupervised learning model mostly involves finding some distance between the input features. Such distance value between features can be considered as the learning value in unsupervised learning models. Below are some learning values used in unsupervised learning.

B) Learning Values

i) Euclidian distance
ii) Manhattan distance
iii) Minkowski distance
iv) Hamming distance
v) Energy value (in ising models)

After defining the learning value of an unsupervised learning model, a training technique is required to optimize this value. From such training, a pattern can be discovered

within the given inputs, under which the input data can be grouped. This training can be generalized as

$$\begin{aligned} &\underset{X,\, d}{\text{optimize}}\ \mathcal{H}(X, d) \\ &\text{subject to } \mathcal{K} \end{aligned} \tag{2.22}$$

where $\mathcal{H}$ is the optimization function (i.e., objective function), $\mathcal{K}$ is the optimization parameter (i.e., constraint), and d is a distance measure between input data.

Many training techniques have been developed to train unsupervised learning models. Similar to supervised learning techniques, the learning techniques in unsupervised learning are also done using conventional optimization methods depending on the nature of the learning value in terms of its convexity, continuity, and linearity. Below are some learning techniques used in unsupervised learning.

C) Learning Techniques

i) Gradient design
ii) Gradient ascent
iii) Variational inference
iv) Nearest neighbors search
v) Gibbs sampling

For a network of non-causal functions, learning techniques such as Gibbs sampling, contrastive divergence, wake-sleep, hopfield learning rule, etc, are used in unsupervised learning network models such as Hopfield networks, Boltzmann machines, restricted Boltzman machines, Helmholtz machines, variational autoencoder, etc, which can all be model using Markov random fields.

2.2.3 Semi-supervised Learning

Semi-supervised learning (also called weakly supervised learning) is a type of machine learning paradigm where the labels are incomplete, inaccurate, and/or inexact. Each of these cases has a different approach to solve. Since the learning and action performance of machine learning models depends on the amount and quality of input (i.e., observation) and/or output (i.e., response) information provided to them, the more input and output information (with high-quality values) available to them, the better their learning and action performance.

Using high-quality labels (i.e., output information) to train a model is called strong supervision. The quality involved the completeness, accuracy, and exactness of the labels. However, having a large dataset with high-quality labels is difficult, leading to different approaches to training models using low-quality labels. Training models using low-quality

labels is called weak supervision. In this supervision, the label can be incomplete, inaccurate, and/or inexact, hence challenging to train with. Different weak supervised learning approaches have been developed depending on the quality of the label.

In the case of incomplete labels, the objective is to train a model from both labeled and unlabeled instances of the training dataset. One technique is to consider the problem as a supervised learning problem for which the dataset is labeled, hence only the labeled instances are used and the unlabelled instances are manually inserted if possible, which may entail huge annotation cost. Another technique is to consider it as an unsupervised learning problem for which the dataset is not labeled, hence only the unlabeled instances are used, and even labels of the labeled instances are removed to increase the size of the unlabeled instances. A common technique is to use a hybrid dataset of both the label and unlabel instances. Machine learning models based on this hybrid technique are what are generally considered semi-supervised learners of incomplete labels.

In the case of inexact labels, the objective is to train a model using imprecise labels, which include misleading labels. A common technique used for this case is to select the precise labels and use them to approximate the exact label of a single or bag (i.e., group) of inexact instances. Using the labels of key instances to label a bag of instances used during learning is commonly called multi-instance learning. During such a learning approach, the bag generator specifies how many instances should be in each bag. A bag can be a text document, an image, a set of stock records, and so on.

In the case of inexact labels, the objective is to train a model using imprecise labels, which include misleading labels. A common technique used for this case is to select the precise labels and use them to approximate the exact label of a single or bag (i.e., group) of inexact instances. Using the labels of key instances to label a bag of instances used during learning is a learning approach commonly called multi-instance learning. During such a learning approach, the bag generator specifies how many instances should be in each bag. A bag can be a text document, an image, a set of stock records, and so on.

In the case of inaccurate labels, the objective is to train a model using wrong labels, which include mislabeled instances. A common approach used is data editing, where a graph of relative neighborhoods is constructed, with each node being an instance, and an edge connects two nodes of different labels. An instance (node) is considered suspicious if it is connected to many edges. This suspicious instance is then removed or re-labeled according to the majority agreement for all annotators.

Since the issue of incomplete, inexact, and inaccurate labels occurs in many machine learning problems such as text classification, image classification, object recognition, object localization, spam detection, stock price prediction, and so on, a semi-supervised learning approach, rather than a supervised or unsupervised learning approaches, can be used in these cases. Some common semi-supervised learning approaches used especially in text classification include the latent variable approach, embedding-based approach,

and hybrid approach. Some common models used for semi-supervised learning include variational autoencoder, restricted Boltzman machines, etc.

Generally, semi-supervised learning can be modeled as supervised learning with noisy (i.e., low-quality) labels. In supervised learning, a dataset $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ is given, where each x_i is a d-dimensional input (or feature vector), y_i is the corresponding high-quality output (or label) define under the space S, and D is assumed to be drawn from some unknown but high-quality distribution P, that is, $(x_i, y_i) \sim P$. In semi-supervised learning, a dataset $\ddot{D} = \{(x_1, \ddot{y}_1), \ldots, (x_n, \ddot{y}_n)\}$ is given, where each x_i is a d-dimensional input, $\ddot{y}_i$ is the corresponding low-quality label define under the space $\ddot{S}$, and $\ddot{D}$ is assumed to be drawn from some unknown but low-quality distribution $\ddot{P}$, that is, $(x_i, \ddot{y}_i) \sim \ddot{P}$.

The objective of semi-supervised machine learning is to find a function $\ddot{f} : \mathbb{R}^d \rightarrow \ddot{S}$, such that for every input-output value pair,

$$\ddot{f}(X) \sim f(X) = y \tag{2.23}$$

where $X = (x_1, x_2, ..x_n)$ is a vector of input values, x_i is the feature vector of the ith input example, $\mathbb{R}^d$ is the d-dimensional input feature space, y_i is the feature vector of the ith high-quality output example, $\ddot{y}_i$ is the feature vector of the ith high-quality output example, n is the size of the dataset, f is the function defining a model using high-quality datapoint D, and $\ddot{f}$ is the function defining a model using low-quality datapoint $\ddot{D}$.

Different functions are used to define $\ddot{f}$. In general, $\ddot{f}$ can be defined as a composite function of supervised and unsupervised actions, f and $\tilde{f}$, respectively, i.e., $(f, \tilde{f}) \sim \ddot{f}$. In this way, the action functions for semi-supervised learning can be any logically sound combination of supervised and unsupervised actions already discussed in the previous sections. This combination can be sequential or parallel depending on the purpose of the semi-supervised action. If the semi-supervised action is causal, then the supervised action is used at the end of the sequence to make a prediction on the real output, whereas if the semi-supervised action is non-causal, then the unsupervised action is used at the end of the sequence to generate an abstract output. For example, using a Restricted Boltzmann machine for semi-supervised learning of incomplete labels, the entire inputs are trained using an unsupervised approach, and the abstract output is then used as input to train a supervised action that makes predictive action on the label instances.

Similar to supervised and unsupervised learning, a semi-supervised learning action requires optimization during the learning process. This also entails a combination of supervised and unsupervised learning techniques in most cases. Hence, a combination of prediction error for the supervised part and distance measures for the unsupervised part are used as learning values during most semi-supervised learning approaches. These learning values are similar to those presented in the previous sections. Using this combination technique, one can define a semi-supervised learning process as an optimization process

of two learning values based on error and distance measures as follows,

$$\begin{aligned} &\underset{X,\ddot{y},d}{\text{optimize}}\ \mathcal{H}(X,\ddot{y},d) \\ &\text{subject to } \mathcal{K} \end{aligned} \tag{2.24}$$

where $\mathcal{H}$ is the composite optimization function based on a combination of error and distance functions, $\mathcal{K}$ is the optimization parameter (i.e., constraint), $\ddot{y}$ is the noisy label, and d is a distance measure between input data.

2.2.4 Self-supervised Learning

Due to the challenges faced in solving the problem of low-quality or unannotated labels in Self-supervised and unsupervised learning, the idea of inter-training available input features became prominent, leading to the idea of self-supervised learning. Self-supervised learning is a machine learning paradigm where the model is trained to learn one part of the input from another part of the input. It actually involves the identification of any unknown input from any known input.

This paradigm is important in machine learning problems where it is mostly impossible and costly to have label data. This is common in natural language processing during the training of large language models (LLM), and in computer vision during the processing of a large number of image frames. In natural language processing, if we have a few words, using self-supervised learning we can complete the rest of the sentence. Similarly, in a video, we can predict past or future frames based on available video data.

Self-supervised learning is advantageous in all these cases because training models on data with low-quality or unavailable labels is more cost and time-effective, as one does not need to annotate and label the data. It achieves this by using the structure of the data to make use of supervisory features within the data sets without depending on labels. However, since, models can be trained much faster when actual label samples are provided, in self-supervised learning the model needs to make sense of the provided unlabeled data, and also generate the corresponding labels. This leads to a high computational cost.

Self-supervised learning is generally considered a supervised learning problem in which the inputs and outputs can be obtained from the data itself, without needing any human annotation or labels. In this way, a self-supervised action can be defined as a supervised action over the input space. Using only the input space reduces its accuracy if tested on actual labels. The quest to increase accuracy has led to the development of more robust learning approaches, which may be different from those used in the other conventional learning paradigms. While most models in the other learning paradigms use a representation-based and/or energy-based approach to learning, self-supervised learning mostly uses a context-based learning approach, alongside representation-based and/or

energy-based approaches.

Examples of models that use context-based learning to solve self-supervised learning problems are the transformer models such as Bidirectional Encoder Representations from Transformers (BERT). Another technique to execute self-supervised learning is to use the energy-based learning approach which is an unsupervised and semi-supervised learning. A hybrid method with different approaches can also be used in the form of a joint embedding architecture.

2.2.5 Reinforcement Learning

Reinforcement learning [18] is a learning paradigm where the learning model, also called an agent, learns by taking actions in an environment and receiving feedback from this environment. It is actually different from supervised and unsupervised learning in that the data is not provided as a fixed set of examples. It is actually a type of interactive learning.

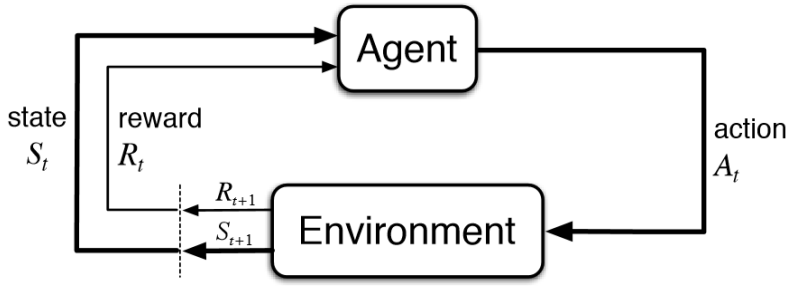


Figure 2.3: Reinforcement learning process.

The idea of reinforcement learning is to learn a target in an environment without any given predefined input data. The agent is allowed to explore and exploit the environment in which the target is assumed to exist. It is during such explorative and exploitative experiences (interaction) with the environment that the agent generates input data to take actions that will lead it to achieve the target. Thus, the reinforcement learning problem involves an agent exploring and exploiting an unknown environment to achieve a target.

In most cases, the agent knows its target but does not know which input to select and learn in order to reach the target. In other words, the agent needs to build a strategy to achieve its target. This strategy involves, selecting the right input and taking the right actions that will connect that input to the desired target. This is similar to the concept of Game theory and Control theory, where agents know their target but need to build a strategy to achieve the target. That is why reinforcement learning is highly used in Game and Control system developments. In some rare cases, the agent is not given any predefined target but is allowed to search the environment for any target it may wish to achieve or master. Even though this is recently not popular, such a paradigm may lead to self-deterministic and self-sufficient agents, especially with an artificial general intelligence.

As shown in Figure 2.3, for any given environmental state S_i, a reinforcement learning

agent receives a reward R_i from the environment for any action A_i it takes. The ultimate goal of all reinforcement learning agents is to optimize this reward and to do so at a lesser cost. To achieve this, it focuses on its action strategy by defining and learning an optimal policy that will optimize the expected reward. Thus, a reinforcement learning algorithm can generally be defined as follows:

$$\max_{\theta} \ \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=1}^{T} r(s_t, a_t) \right] \tag{2.25}$$

$$\text{subject to } \ p_\theta(\tau) = p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t) \tag{2.26}$$

where $\tau = (s_1, a_1, \ldots, s_T, a_T)$, s is the environment state, a is the action state, t is the number of state-action pair, $r(s_t, a_t)$ is the reward, $\pi_\theta(a_t|s_t)$ is the policy, and $p(s_{t+1}|s_t, a_t)$ is the model.

From the equations, rewards depend on the policy and the model (system dynamics, environment, etc). In general, reinforcement learning algorithms can be broadly categorized as model-free and model-based. A model in this case represents the environment and it is defined by the transition probability distribution (or transition model) and the reward function. The most common transition logic used to define the environment is the Markov decision process (MDP). The main distinction between model-free and model-based reinforcement learning is that in model-free reinforcement learning, the model is ignored and the reward depends on the policy, whereas in model-based learning, a cost function is usually defined, which is then used to calculate the optimal actions using the model directly without a policy.

Furthermore, model-based reinforcement learning algorithms build a model of the environment by sampling the states, taking actions, and observing the rewards. For every state and possible action, the model predicts the expected reward and the expected future state. They seek to understand the environment by creating a model of the environment based on their interactions with the environment. This environment is either given to them or they are required to learn the environment. This leads to two types of model-based reinforcement learning algorithms; model-given (e.g., AlphaZero) or model-learning (e.g., world models, model-based priors for model-free, and model-based value expansion).

On the other hand, model-free algorithms seek to understand their environment by learning the consequences of their actions through experience in the environment. They do not build an explicit model of the environment. Model-free algorithms can be divided into value-based (e.g., SARSA and Q-learning models), policy-based (e.g., Monte Carlo policy gradient models such as REINFORCE, and deterministic policy gradient models), or hybrid (e.g., actor-critic model). Value-based algorithms use optimal policy which is a di-

rect result of estimating the value function of every state accurately. This value function is mostly defined using the Bellman equation. Policy-based algorithms, on the contrary, directly estimate the optimal policy without modeling the value function.

2.3 Machine Learning Approaches

Machine learning approaches define the techniques used to implement a machine learning task. Many different machine learning approaches exist in the literature which are used by different models. These machine-learning approaches can be grouped into different categories as described below.

2.3.1 Categorization of Learning Approaches

i) Number of inputs:

A machine learning model can use one or more input instances during the learning of the output. In this category, the model can use more inputs for fewer outputs, equal input and output, or fewer inputs for more output instances. The first case consists of multi-instance learning approaches. The second case consists of stochastic (i.e., online) learning and batch learning approaches in passive mode. The third case can be used during active learning.

ii) Nature of inputs:

The nature of the input in terms of value type e.g., continuous or discrete input, does not really play a great role in categorizing machine learning approaches but it is an important factor that affects the machine learning performance.

iii) Number of outputs:

The number of output variables and instances affects the performance and design of a machine learning model. Two common learning approaches in this category are multi-target (or multi-label) learning and multi-class learning.

iv) Nature of output:

The type of value at the output greatly influences the design of machine learning models. The output value can be continuous or discrete, and static or dynamic. Discrete output problems are solved using classification learning approaches while continuous output problems are solved using regression learning approaches. In most cases, the values of the label are static over the label space, however, these values can also change over the label space, in which case a multi-label learning approach can be used to address the change in output value across the different output domains.

iv) Number of actions:

The number of actions a model can take simultaneously on a target is an important factor used to categorize machine learning approaches. If more than one action (e.g., task)

can be done simultaneously by the agent, then a multi-task learning approach is used, otherwise, a single-task learning approach is adopted.

v) Nature of the action:

This consists of the underlying logic of the action. This logic can be inductive, deductive, transductive, abductive, and so on. For each logic, a different learning approach can be adopted such as inductive learning, deductive learning, abductive learning, transductive (e.g. transfer) learning, and so on.

vi) Number of agents:

The number of agents (or action nodes) in a given model is an important factor that influences model output. There can be single-agent or multi-agent models, leading to different learning approaches. The most important models in this categorization are those related to multi-agents, in which case we can use a variety of learning approaches such as ensemble learning, federated learning, deep learning, and so on. In this dissertation, we proposed a multi-agent learning approach called collaborative learning, alongside models and their applications based on this approach.

vii) Extrinsic property:

This consists of external properties of a model that define their action and learning process. In this category, two main approaches can be distinguished; parametric and non-parametric. A parametric approach can be used to design and learn a model, where the model learns based on well-defined parameters. A non-parametric approach does not require parameter learning.

viii) Intrinsic property:

These are properties that are used by a model to abstract meaning from the input information during learning and action. The main objective is to learn the meaning embedded in the data and to achieve this, different abstraction techniques are used such as representation, energy, and context. In this category, many approaches can be used but the most common are representation-based learning, energy-based learning, and context-based learning. These are very common in multi-agent learning approaches. Other important intrinsic properties are the discriminative and generative properties associated with machine learning models. Energy-based models are generally generative by design, while representative-based models mostly use a discriminative approach. Depending on the distribution (or spread) of this intrinsic property across multi-agents, different categorizations can be used such as flow-based and diffusion-based, which can be used in both discriminative and generative models.

In the next sub-sections, I will focus on multi-agent learning approaches as it is under which this dissertation can be categorized. This will include deep learning, ensembling learning, and federated learning which are conventional approaches, and my proposed collaborative learning approach. In each case, different related categorizations such as the

intrinsic property used by their models will be discussed, together with common models based on the approaches.

2.3.2 Deep Learning Approach

Deep learning is a machine learning approach used by a human-brian-inspired multi-agent model called neural networks. A deep learning neural network consists of interdependent nodes (learners) that collectively act, sequentially and parallelly, on a target.

Based on the intrinsic property used in the neural network, different neural network models have been developed. This includes representative-based neural networks such as Deep feedforward neural networks (DNN), Convolution neural networks (CNN), recurrent neural networks (RNN), and recursive neural networks (RvNN) models; energy-based neural networks such as Hopfield networks, Boltzmann machine (BM), and Restricted Boltzmann machine (RBM); context-based neural networks such as transformers models like Bidirectional Encoder Representations from Transformers (BERT) and Attention-based bidirectional CNN-RNN deep model (ABCDM) models.

Furthermore, deep learning approaches can be used to achieve the objective in any learning paradigm. In general, energy-based deep learning models are used for unsupervised learning tasks, representation-based deep learning models are used for supervised learning, and context-based deep learning models are used for a hybrid of supervised and unsupervised learning tasks such as semi-supervised and self-supervised learning tasks.

The learning techniques used in the different deep learning approaches also differ. Reppresentation-based deep learning models use backpropagation learning techniques. energy-based deep learning models use a type of backpropagation called contrastive divergence. Context-based deep learning models use the attention mechanism.

2.3.3 Ensemble Learning Approach

Ensemble learning is a machine learning approach based on the aggregation of the actions of multiple independent learners, sequentially or parallel, on a common target. Ensemble learning is mostly used on causal multi-agent models such as predictive models and supervised learning models, to obtain better action performance on a target. However, they can also be used in non-causal models such as descriptive and unsupervised learning models.

There are different types of techniques used to implement and distinguish ensemble learning models. Related to their method of training, three main techniques can be distinguished; bagging, boosting, and stacking. They form the three main ensemble learning models. However, ensemble learning can also be distinguished based on their action aggregation techniques. Related to the aggregation method of their action, many techniques can be used such as averaging, voting, and combination, leading to averaging ensemble, voting ensemble, and combinatorial ensembling.

unlike deep learning which uses interdependent nodes (learners), ensembling learning uses independent learners. Also, ensemble learning involves the aggregation of the actions of each learner on the target, while deep learning involves the collective integration, without aggregation, of the actions of each node in the neural network. The agents in a deep neural network learn collectively in an interdependent manner, while those in ensembling learning learn independently from each other. However, an ensemble learning model can be built based on independent deep learning models. These are commonly called deep ensemble learning models. Another popular implementation of ensemble models is an ensemble of decision tree models, commonly called random forest.

2.3.4 Federated Learning Approach

Federated learning is a machine learning approach based on the aggregation of the parameters of multiple independent distributed learners on a common target. The distributed architecture can be centralized or decentralized over the input space, and each learner is fed with input data from a different source, defined either by a set of segmented observations or segmented features.

Due to these, federated learning can be distinguished based on the segmentation technique of the input space. If the input space segmentation is based on the input observation the federation is considered a horizontal federation, and if the input space segmentation is based on the input features the federation is considered a vertical federation. More so, depending on the structural organization of the learners, federated learning can be centralized or decentralized. Furthermore, federated learning models can be categorized based on the aggregation technique of the federated parameters. These include federated averaging, dynamic regularization, dynamic aggregation, and stochastic gradient descent.

Unlike ensembling learners that aggregate action values, federated learners aggregate parameters. Hence federated learning can only be used on parametric models, whereas ensemble learning can be used on both parametric and non-parametric models. Also, the fact that federated learning permits only the shearing of model parameters between the agents in a decentralized architecture or between the clients and server in a centralized architecture, rather than shearing actions and input domain values, they are preferable in situations where data privacy, security, and access across heterogeneous environments are required. This is possible because of its use of feature-based (or vertical) segmentation of input data, each forming a different input space associated with a single local agent.

However, both federated and ensemble models, together with deep learning models, focus on a common target but use different collective approaches to achieve the target. Also, similar to ensemble learning, deep learning neural networks can be used to build each federated learning agent, but the reverse is unlikely. This implies that the deep learning model is a fundamental multi-agent learning model that can be integrated into the

other multi-agent models. This is simply because of the high cohesiveness and interdependencies between nodes in a deep-learning neural network, making it difficult to disintegrate without obstructing the entire principle that binds them.

2.3.5 Multi-agent Reinforcement Learning Approach

Multi-agent reinforcement learning (MARL) approach involve the studying of the behavior of multiple learning agents that coexist in a shared environment using reinforcement learning paradigm. In this approach, each agent is motivated by its own rewards, and acts to advance its own interests. In some environments these interests are opposed to the interests of other agents, resulting in complex group dynamics. Due to this, MARL structure can be categorized into cooperation, competition, or a hybrid of cooperation and competition. Designing an MARL will require one of these settings.

The main logic that govern the interaction between MARL agents is based on game theory. Game thoery actually defines a type of logical rules of cooperation and competion between rational agents and it is based on these rules that the interactions between agents in MARL are built. However, the actual operations of each agent and their rewards is based on reinforcement learning paradigm. The MARL approach has high application in robotocs. Examples of models built using MARL approach include Deep Blue and AlphaGo.

2.3.6 Probabilsitic Learning Approach

Probabilistic learning is a learning approach where the model makes predictions based on probability distributions of the possible outcomes. This is one of the fundamental machine learning approaches on which most data-driven learning approaches and models inherit their design and operational concepts. In fact, the learning and actions of all machine learning models can be described using probabilistic logic, making probabilistic models a base model in machine learning.

One of the limitations that prevented their use in diverse applications was their computational cost with respect to the dimensionality of the input domain. The more input features, the more complex it becomes to estimate the marginal distribution, which is a key distribution in most probabilistic models. However, there have been advances in their field of probabilistic learning and many probabilistic models can be used to perform more complex tasks like image recognition. This is also due to the increase in computational power of computing devices on which machine learning models are executed.

One key issue with probabilistic models is that their networks can be modeled using the graph theory, where some can be modeled based on directed graphs like Bayesian networks, or undirected graphs like Markov networks. This has led to a branch of machine learning called probabilistic graphical models (PGM), with the Bayesian and Markovian networks forming the two main branches of PGM. The learning mechanisms of these

networks are basically different from those of deep learning, federated learning, and ensemble learning. A common learning technique is belief propagation based on the sum-product rule, average-product rule, and so on.

However, more variants of probabilistic models have emerged with interesting results such as variational inference models, leading to the popular variation autoencoder (VAE) model. Also, some variants of PGM use different belief propagation techniques such as the energy-based belief propagation technique which is based on a probabilistic measure, specifically Gibbs measure (or energy), commonly used for training energy-based neural networks such as Hopfield networks and Boltzmann machine. A common and less computationally expensive variant of PGM is a Naive Bayesian network, which is based on the assumption of statistical independence between the input features to limit the complexity of estimating the marginal distribution.

2.3.7 Collaborative Learning Approach

Literally, collaboration is the process where two or more entities work together, with the same or different purpose, to complete a common action (e.g., mission or task) or achieve a common target (e.g., vision, goal, objective, outcome, or output). Thus, agents can collaborate on a target or on an action, but since target and action are linked by a purpose, the collaboration on an action entails a collaboration on a target, and vice-versa. In this way, a collaborative action on a target may be divided into multiple partial actions defined by the nature of the target perceived by the agent. Collaborative learning is an approach to machine learning where partial actions of local agents are learned collectively, in sequence and/or parallel, to achieve a common target.

Based on this definition, one can consider collaborative models as a generalization of all multi-agent models that are designed to act on or work toward a common target. These include deep learning models, ensembling models, and federated probabilistic models. In this dissertation, we use classification as the partial action on the target, to achieve a collaborative classification on the target. However, partial regression and classification can also be used collectively. As I will discuss in future chapters, this proposed concept of collaborative models will be applied to both generative and descriptive modeling, together with their corresponding collaborative learning techniques.

2.4 Data Heterogeinety

Heterogeneity is a dissimilarity between elements that comprise a whole, and it is the opposite of homogeneity, which implies similarity between elements that comprise a whole. A heterogeneous dataset is any dataset in which there is high variability of data types and formats and is usually generated from different sources. When heterogeneity is present

in the dataset, there is likely a diversity in the different features and observations of the dataset.

2.4.1 Sources and Implications of Data Heterogeneity

Usually, heterogeneity in a dataset is caused by noise resulting from missing values, data inconsistency, ambiguity, and redundancy. Due to the presence of noise, using such a dataset for training a model will definitely limit the performance of the model. Such types of noisy data are often generated from large-scale data sources such as Internet of Things (IoT) networks and large language models.

Depending on the sources that generate the data, data heterogeneity can generally be divided into four types [5]: syntactic, conceptual, terminological, and semiotic. Syntactic heterogeneity occurs when two data sources are not expressed in the same language. Conceptual heterogeneity also known as semantic heterogeneity or logical mismatch, denotes the differences in modelling the same domain of interest. Terminological heterogeneity stands for variations in names when referring to the same entities from different data sources. Semiotic heterogeneity also known as pragmatic heterogeneity, stands for different interpretations of entities by people.

The presence of heterogeneity in data has both negative and positive implications depending on the type of heterogeneity. The presence of different features generally enriches a model's knowledge base about a problem, leading to a more robust or generalized performance. However, the increase in the number of features can instead lead to the curse of dimensionality. Also, the presence of heterogeneity in value dispersion such as large differences between values of the same observation or feature, that is the presence of overlayers, can lead to poor model accuracy.

Eliminating these different types of heterogeneity is an important part of data preprocessing in machine learning. Different techniques have been developed to reduce heterogeneity in data.

2.4.2 Techniques to reduce Heterogeneity in Data

The main goal of data heterogeneity reduction techniques is to create homogenous data. Many techniques have been developed to reduce data heterogeneity in order to facilitate data analysis and machine learning tasks. These techniques include data normalization, standardization, and data imputation.

Data normalization is a technique used to create homogeneity between data instances. It applies scaling over the instances of a dataset. This technique is required when the data distribution is unknown or doesn't follow a Gaussian distribution. Different scaling techniques used include min-max scaling, z-score scaling, and log scaling.

Data standardization is a technique used to create homogeneity between data attributes,

hence used for multivariate data and multivariate analysis. It mostly involves the use of scaling and encoding techniques to harmonize the features, i.e., render them homogeneous. Common scaling techniques for quantitative data usually consist of normalization scaling techniques such as min-max scaling, z-score scaling, etc. Common encoding techniques for categorical data include one-hot encoding, label encoding, frequency (or count) encoding, and binary encoding.

The data imputation technique consists of using alternative values in place of missing data. It is referred to as unit imputation when replacing a data point and as item imputation when replacing a constituent of a data point. This technique is employed because it would be impractical to train a model or carry out data analytics with missing data. Common data imputation techniques include maximum/minimum value, linear/average interpolation, previous/next value, most frequent value, mean/median value, value prediction, and K-nearest neighbor techniques.

In this study, apart from missing value cases, the problem of heterogeneity in data is handled by segmenting the data across features i.e., vertical/feature segmentation, or across instances, i.e., horizontal/instance segmentation, using measures such as distance and similarity measures. Features and instances that are far apart or more dissimilar are separated and those that are close or more similar are grouped together. Using this approach, we then deploy different agents to learn and carry actions on the data using the collaborative framework proposed in this study.

2.5 The Curse of Dimensionality

Technically, in a data-driven analytical process, such as data-driven machine learning, in order to obtain a reliable statistical result, the amount of data instances (i.e., observations) need to grow exponentially with dimensionality (i.e., the number of features). So, if the dimensionality increases without an increase in data instances, the available data becomes sparse and dissimilar, and hence unreliable to use in obtaining reliable statistical results such as training and predictive accuracy of a data-driven machine learning models. This phenomenon is called "the curse of dimensionality" [3, 4].

2.5.1 Origin and Implications of the Curse

The curse of dimensionality of data for a given model and process is related to the correlation between the features of the data, amongst others. The more features contained in a dataset the more correlation exists between the features, which impedes the learningability of most data-driven models such as regression models, reducing their performance. This also occurs in non-linear models such as multilayer perceptions due to the presence of multi-collinearity. So, the challenge of most learning models is not only to learn the

target but also to do so with the least possible correlation between the features.

Furthermore, since the increase in the number of features and instances increases intrinsicly the correlation in the data, the curse of dimensionality is considered to be caused extrinsincly by the increase in the dimension of data without an exponential increase in the data instances. However, the proportion of the dimension and instances of data required to avoid such a curse in any given data-driven process and model will depend on the model and process to which the data is to be used, hence the properties of these models, such as their parameters and/or hyper-parameters, should be considered.

In machine learning, studies have been carried out to determine the initial point of such a course for some models and datasets. The result of one of such studies is called the Hughes phenomenon [19]. This phenomenon describes a situation where in a fixed number of training samples, the average (expected) predictive power of a predictor (classifier or regressor) first increases as the number of dimensions or features used is increased but beyond a certain dimensionality, it starts deteriorating instead of improving steadily.

The implication of the curse of dimensionality is a low performance of the model and process to which the data is used. The performance measures affected by such a cursed are mostly quantitative measures for both causal and non-casual data-driven processes and models. These include but are not limited to accuracy, precision, and recall. The relationship between the level of the curse and the performance is an interesting area of concern, together with the quantification of the level of the curse. Furthermore, the high correlation between features leads to high variance in correlation-sensitive models, leading to overfitting. Conversely, since variance and bias are antagonistic, a high model bias implies a low correlation in the data and will lead to underfitting. hence, a bias-variance balance is required and can be achieved using a correlation balance over the model.

2.5.2 Techniques to avoid the Curse

To avoid the curse of dimensionality, one needs to first measure the limit to which a model and process will enter into the curse. In some fields such as combinatorics, where discrete data is mostly used, such a limit for extremely large combinatorial data, i.e., combinatorial explosion, can be determined for any combinatorial process and model to be at the limit $n/2 > n - p$, where n is the sample space (i.e., instances) and p is the number of positions (i.e., features). However, for most data-driven machine learning models, there is a need for more data instances even with reduced dimensions, hence dimension reduction agnostic.

Dimensionality reduction is prolific in machine learning, especially in feature engineering and data preprocessing. Many dimension reduction techniques exist, however, these techniques do not provide the limit of the number of dimensions to be reduced with respect to the number of instances. Common dimensionality reduction techniques include principal component analysis (PCA), independent component analysis (ICA), linear

discriminate analysis (LDA), non-negative matrix factorization (NMF), canonical correlation analysis (CCA), factor analysis, and K-nearest neighbors (k-NN).

In this study, rather than using dimensionality reduction for curse avoidance, we used dimensionality segmentation, commonly called feature segmentation. High-dimensional datasets are randomly or based on a measure, segmented into groups. Then each of these segmented datasets is given to separate models to collaboratively act and learn on a target. Common segmentation measures include similarity measures such as correlation coefficient and distance measures such as Euclidian distance. This enables us to achieve a balanced correlation distribution over the partial models in the collaboration. Hence, solving the bias-variance balance problem in model learning and generalization.

Chapter 3
Modeling

Modeling collaborative learning agents in a heterogeneous environment entails defining their prediction and learning actions, source data, and target. Conventionally, some machine learning models can be considered as a type of collaborative learning model because they consist of multiple models (agents) working together to solve the same problem. These models include artificial neural networks (ANN), deep neural networks (DNN), deep belief networks (DBN), Bayesian networks (BN), Markov networks (MN), generative adversarial networks (GAN), restricted Boltzmann machines (RBM), deep reinforcement learning (DRL), ensemble learners (EL), and federated learners (FL).

These models can be categorized as causal, non-causal, or hybrid. ANN, DNN, BN, EL, and FL are usually defined with causal actions to achieve their target, RBM and MN use non-causal actions, while DRL, DBN, and GAN use both causal and non-causal actions. In general, causal models are commonly used for supervised learning while non-causal models are used for unsupervised learning, a hybrid model of causal and non-causal usually has both supervised and unsurprised learning properties.

In this study, we proposed a collaborative learning model with both causal and non-causal actions. Probabilistic logic is used to prove the correctness of the model and the presence of its casual and non-causal parts.

3.1 Conventional Collaborative Models

Amongst the conventional models, the DBN [20], which is a hybrid model with casual and mutual actions, is presented below as a comparison with the proposed model.

3.1.1 Deep Belief Network

As shown in Figure 3.1, a DBN is a type of deep learning algorithm that uses RBM [21] at its input and hidden layers and an output layer based on a supervised learning model such as logistic regression for classification or regression. Since the RBM is an unsupervised generative model, a probability distribution is learned over the input data by each RBM.

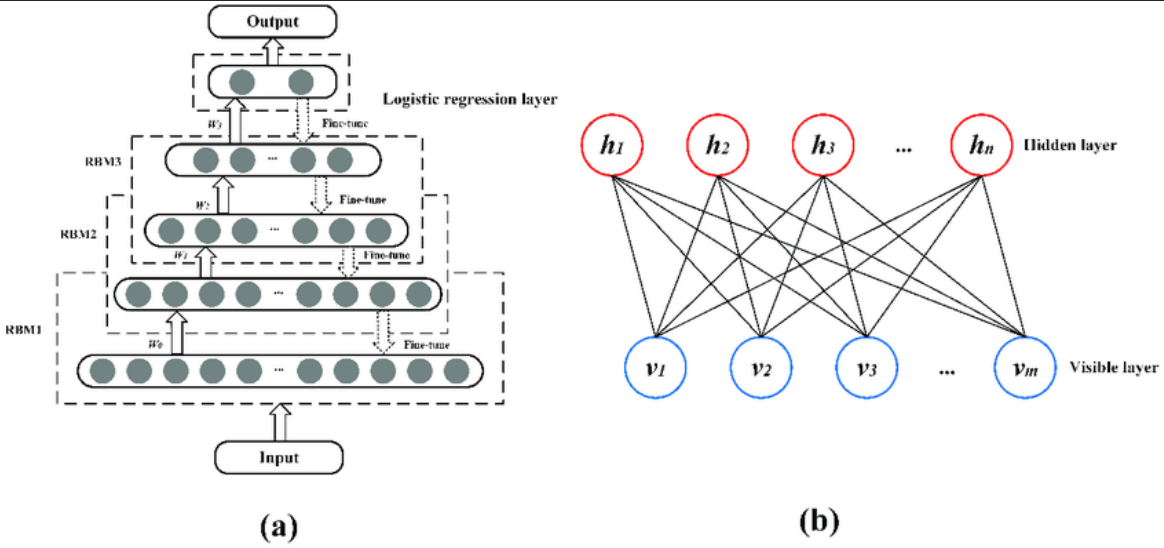


Figure 3.1: DBM architecture with RBM and Logistic regression layers.

Defining a DBN involves defining both an RBN and a supervised learning model such as logistic regression. The non-causal action of a DBN is executed by an RBN and is defined as follows:

$$p(v) = \frac{1}{Z} \sum_{h} exp(-E(v,h)) \tag{3.1}$$

where v is a visible layer to h, h is a hidden layer to v, Z is the partition function (used for normalizing), and $E(v,h)$ is the energy function assigned to the state of the network.

A lower energy indicates the network is in a more stable state and hence in a desirable configuration. So, the learning objective of the non-causal part (defined by an RBN in this case) is to reduce the energy state of the RBN model by maximizing the expected value of the non-causal action $p(v)$ over the visible states v. This can be represented as

$$\underset{w}{\text{argmax}} \ \mathbb{E}_v[\log p(v)] \tag{3.2}$$

The learning process of the RBN can be done using different techniques such as contrastive divergence [22] with weight updates using gradient descent, defined as follows:

$$w_{ij}^{(t+1)} = w_{ij}^{(t+1)} - \eta \frac{\partial \log p(v)}{\partial w_{ij}} \tag{3.3}$$

where w is the weight of the RBM model, and η is the learning rate.

At the output layer, a logistic regression model can be used to execute a causal action.

Such layer can be defined using a causal function as follows:

$$\hat{h}_i = p(h|v;\theta) \tag{3.4}$$

where θ is the weight of the logistic regression function, $\hat{h}_i$ is the causal (predicted) action on the output target h.

Learning such a logistic regression function can done directly by applying a gradient descent optimization on a lost function such as cross-entropy, mean square error, or mean absolute error loss functions. The learning process can be defined as follows:

$$\hat{h}_i^{(k)} = f(v_i;\theta^{(k)}) \tag{3.5}$$

$$Z_i^{(k)} = \Gamma(h_i,\hat{h}_i^{(k)}) \tag{3.6}$$

$$\theta^{(k+1)} = L(Z_i^{(k)}) \tag{3.7}$$

where h is the true output value, h is the predicted output value, θ is the optimized parameter, $L(Z)$ is an abstraction of the learning operation of the model, Z is the learning value (i.e., the lost value), $\Gamma(h,\hat{h})$ is the function (i.e., the lost function) defining the learning value, and the superscript k in each variable represents the learning epoch.

During the training of a DBN, the output of one trained RBM is used as the input for the subsequent stacked RBM, and so on, up to the level of the output layer, which uses the output of the previous RBM to make predictions on the real target, especially after the RBM part have been trained. In reality, the RBM layers operate as unsupervised layers, with no real specification of their target, and the output layer then enables the DBN to focus on a specific target using supervised learning on the output of the RBM.

One of the main advantages of DBN is its ability to learn features from the input data in an unsupervised way, and then use these features to make specific decisions on a given target (task). The number of unsupervised learning layers added to a DBN increases its complexity and performance, making it highly applicable to image classification, where the lower layers can pick up on fundamental details like edges of the image and the later layers can then learn more intricate properties, such as forms and objects in the image.

3.2 Proposed Models

The proposed models are based on a proposed collaborative framework defined and proven using probabilistic logic. However, other logic can be used to proof the same collaborative theory. This theory is based on two main action types, causal and mutual (non-causal), that are used by agents from different input domains, to collaborate with each other over a target.

3.2.1 General Framework

The collaborative theory can literally be stated as follows;

"Assuming the input domain of each action/agent is independent of the others given a target, the collaborative causal action of the collection of mutually related actions/agents on the target is the integration/aggregation of their partial causal actions".

This is mathematically represented using probabilistic logic as follows;

Axiom 3.2.1 (Conditional independence of input properties)

$$P(X_i|X,y) = P(X_i|y) \tag{3.8}$$

Proposition 3.2.1

$$\hat{y} = P(y|X) = \frac{1}{P(y)^{n-1}} \prod_{i=0}^{n} P(y|X_i) \left(\frac{P(X_{i+1}| \bigcap_{\mu=0}^{i} X_\mu)}{P(X_{i+1})} \right)^{-1} \tag{3.9}$$

where y is the given set of categories, X is a vector of feature vectors $X = (X_0, X_1, \ldots, X_n)$ and $X_i = (x_1, x_2, \ldots)$, $\hat{y}$ is the posterior distribution (i.e., class posterior) of y given X, $P(y)$ is the prior distribution (i.e., class prior) of y based on X, $P(y|X_i)$ is the partial posterior distribution (i.e., partial class posterior) of y given X_i, $P(X_{i+1})$ is the prior distribution (i.e., observation prior) of X_{i+1} based on $\bigcap_{\mu=0}^{i} X_\mu$, $P(X_i| \bigcap_{\mu=0}^{i} X_\mu)$ is the posterior distribution (i.e., observation posterior) of X_i given $\bigcap_{\mu=0}^{i} X_\mu$, and n is the number of features vectors.

The causal and mutual action can be distinguished in the formula as follows;

$$C_i \triangleq P(y|X_i) \quad \text{(causal action)} \tag{3.10}$$

$$M_i \triangleq \frac{P(X_i|X_\mu)}{P(X_i)} \quad \text{(mutual action)} \tag{3.11}$$

where y is the target (or output) property, X_i is the vector of input properties of action i, C_i is the causal action of agent i on the target, M_i is the mutual action of agent i with respect to neigbhoring agents, and $P(.)$ denotes a probability function.

Applying such representation to proposition 6.2.1 will lead to

$$C = \frac{1}{P(y)^{n-1}} \prod_{i=0}^{n} C_i M_i \tag{3.12}$$

For clarity, the proof of proposition 6.2.1 is provided below.

Consider the joint probability distribution $P(X_1, X_2, X_3, y)$.

$$P(X_1, X_2, X_3, y) = P(X_1)P(X_2|X_1)P(X_3|X_1, X_2)P(y|X_1, X_2, X_3) \tag{3.13}$$
$$P(X_1, X_2, X_3, y) = P(y)P(X_1|y)P(X_2|X_1, y)P(X_3|X_2, X_1, y) \tag{3.14}$$

Equating (A1) and (A2), -0cm

$$P(y|X_1, X_2, X_3) = P(y)P(X_1|y)P(X_2|X_1, y)P(X_3|X_2, X_1, y)\frac{1}{P(X_1)P(X_2|X_1)P(X_3|X_1, X_2)} \tag{3.15}$$

Applying Axiom 6.2.1,

$$P(y|X_1, X_2, X_3) = P(y)P(X_1|y)P(X_2|y)P(X_3|y)\frac{1}{P(X_1)P(X_2|X_1)P(X_3|X_1, X_2)} \tag{3.16}$$

Applying Bayes rule to $P(X_1|y)$, $P(X_2|y)$, and $P(X_3|y)$

$$P(y|X_1, X_2, X_3) = P(y|X_1)P(y|X_2)P(y|X_3)\left[\frac{P(X_2|X_1)P(X_3|X_1, X_2)}{P(X_2)P(X_3)}\right]^{-1}\frac{1}{P(y)^2} \tag{3.17}$$

Therefore, for $P(y|X = X_0, X_1, X_2, X_3, \ldots, X_n)$

$$P(y|X) = \frac{1}{P(y)^{n-1}}\prod_{i=0}^{n} P(y|X_i)\left(\frac{P(X_{i+1}|\bigcap_{\mu=0}^{i} X_\mu)}{P(X_{i+1})}\right)^{-1} \tag{3.18}$$

Using this collaborative framework, I have been developing different collaborative machine-learning models for different applications. In this dissertation, the following three collaborative models were developed and their applications will be provided in the next chapters.

3.2.2 Collabo: A Discriminative Collaborative Learning Approach

Collabo is an application of the collaborative framework on target with continuous data and using a discriminative approach. Application of this model and its variants may range from image processing, regression analysis, security detection, etc. In Chapter 4, we present the Collabo model and its application to DDoS detection in medical IoT.

3.2.3 GenCo: A Generative Collaborative Learning Approach

Genco is an application of the collaborative framework on target with discrete data and using a generative approach. Application of this model and its variants may range from spam classification, text classification, security detection, etc. In Chapter 5, we present the GenCo model and its application to text classification in natural language processing.

3.2.4 CoPreMo: A Collaborative Predictive Model in Time Series

CoPreMo is an application of the collaborative framework on target with time series data that can be continuous or discrete. Application of this model and its variants may range from signal processing, target tracking, coding theory, large language modeling (LLM), etc. In Chapter 6, we present the CoPreMo model and its application to target tracking in ADAS/AD vehicles.

3.2.5 Comparison with Conventional Models

The advantage of this model over other collaborative and non-collaborative models is its model diversity to contain other models, its explainability, stability, robustness, generalization ability, and reduced uncertainty as we shall present in the subsequent chapters of this thesis. Similar to most conventional collaborative models, increasing the number of agents in this proposed model will also increase its performance.

However, the computational cost of the model will be higher than most conventional models over the same task if the number of collaborative agents increases. Thus, there is a need for a balance between design, performance, and cost of the model. Such an analysis is reserved for future work.

3.3 Performance Measures

3.3.1 Action (Belief) Performance

Different performance measures can be used to evaluate causal and non-causal actions. Causal actions such as predictions can be evaluated using statistical measures such as accuracy, precision, recall, f-score, and point-based or batch-based prediction error. These measures are usually based on the specificity (true negative rate) and sensitivity (true positive rate) values of the model. In general, these performance values depend on the true output of the target.

Non-causal actions cannot be evaluated using such techniques because non-causal actions do not have any real target to be compared with. In some cases, non-causal models are measured using their descriptive (or explanatory) performance on the task. For example, in the case of clustering algorithms, the number of clusters they can form with respect to the actual number of grouped instances in a dataset. In general, their system performance such as speed, memory, and computational cost is mostly considered.

3.3.2 Learning Performance

The performance of a learning action or algorithm is an important aspect of machine learning. It is generally measured using the learning value (i.e., prediction error) over a

learning period. This is done in order to capture the generalization of the model on the data during training. However, a more formal way is to use the generalization error, which is a combination of the bias error and the variance error.

3.3.3 System Performance

In recent years, there has been an increase in the interest of a system approach to artificial intelligence and machine learning design. This has given rise to the use of system performance measures to evaluate machine learning models. Such measures include throughput, robustness, latency, response time, FLOPs, MOPs, etc.

The framework and models proposed in this dissertation have more to contribute in this direction as it was designed with a system logic in mind, among others.

3.4 Statistical Signifcance

An important process after obtaining results from a model is to check if the results have any statistical significance, that is if the results are not just some random perturbation of the model. Hence, it is the claim that the results of a test or experiment are not explainable by chance alone. The search for statistical significance in a model involves the use of statistical testing techniques to test the results of a model. These results can be the output of the model or its performance results. In general, the concept of statistical significance of a model or result can be used in any situation where results are generated by a model based on random input. The significance process usually involves testing techniques such as the t-test, z-test, and f-test based on the Anova frameworks. The test for model significance is important in model comparison.

3.5 Confidence Interval

It is a range of values that describes the uncertainty surrounding an estimate. A point estimate uses a single value to estimate a population parameter. In contrast, an interval estimate uses a range of values to estimate a population parameter. A confidence interval is a type of interval estimate. It is used as a technique to confirm if the output of the model is a reflection of the input. The degree to which this can be true is considered as the confidence interval of the model.

3.6 Model Selection

Model selection is a way to select a model from other models on a given input and/or output using certain measures. These measures include performance measures, statistical significance, and confidence intervals. However, most of these measures don't take into

account the model parameters during comparison. Models have been proposed in the literature that take into account this property and include the Aikaike information measure (AIC) and the Bayesian information measure (BIC). For a robust selection, it is required to use a combination of multiple comparison approaches.

In this study, the framework and models I proposed can be evaluated using all these performance measures. These measures can be estimated using only the parameters of the framework and models that define their causality and mutuality. This implies they can be used for diverse applications to mimic the behavior and actions of humans, that is to model cognitive agents that can co-exist and adapt in the human society.

Chapter 4
Application 1: Security in Heterogeneous Medical Data using Collabo

4.1 Background

The amount of medical data generated from medical devices has surged in recent years owing to the increase in medical attention and digital requirements imposed by the COVID-19 pandemic [23]. Coupled with the increase in electronic communication due to globalization, this increase in digital requirements also increases the risk of security threats to medical devices as well as their generated data.

Several techniques have been used to implement security solutions for medical devices and data [24]. Generally, the conventional approach uses a deterministic process, based on a set of logical rules. This approach is used in conventional information and cybersecurity software, particularly for signature-based detection. Another approach is the use of a stochastic process, such as a machine learning process.

With the improvement of learning algorithms in recent years, the latter has gained much attention from research and industry, especially in behavioral detection. In some cases, a hybrid approach consisting of both deterministic and stochastic processes is used.

One problem with medical data is that they are generated by devices with different properties, whose values may have little or no correlation with each other; that is, they are highly heterogeneous. Observation from a thermometer and from an Electrocardiogram (ECG), for example, may have little or no correlation with each other, and using one to interpret the other may lead to incorrect medical decisions. Managing such unrelated types of data is conventionally performed using non-relational databases such as big data.

Big data is a database management architecture that enables the management of numerous unrelated datasets from different sources. The main processes of the big data architecture are presented in [25]. These processes are incorporated into many software packages that are used for big data services, such as Hadoop, MongoDB, and HPCC.

For medical big data, data sources mostly originate from medical devices. Once col-

lected from these sources, the data can be processed in batches or streamed. Analytics can be performed on the data to enable decision-making. For remote medical devices, there is a high requirement for their data to be collected and transferred to a centralized server that hosts the big data database, where most of the analytics is performed.

To support the high requirement for the transfer of medical data from one location to another, there is a need for an efficient communication network to interconnect the devices that generate the data. This requirement to interconnect medical devices has led to the use of Internet of Things (IoT) technology in most medical institutions.

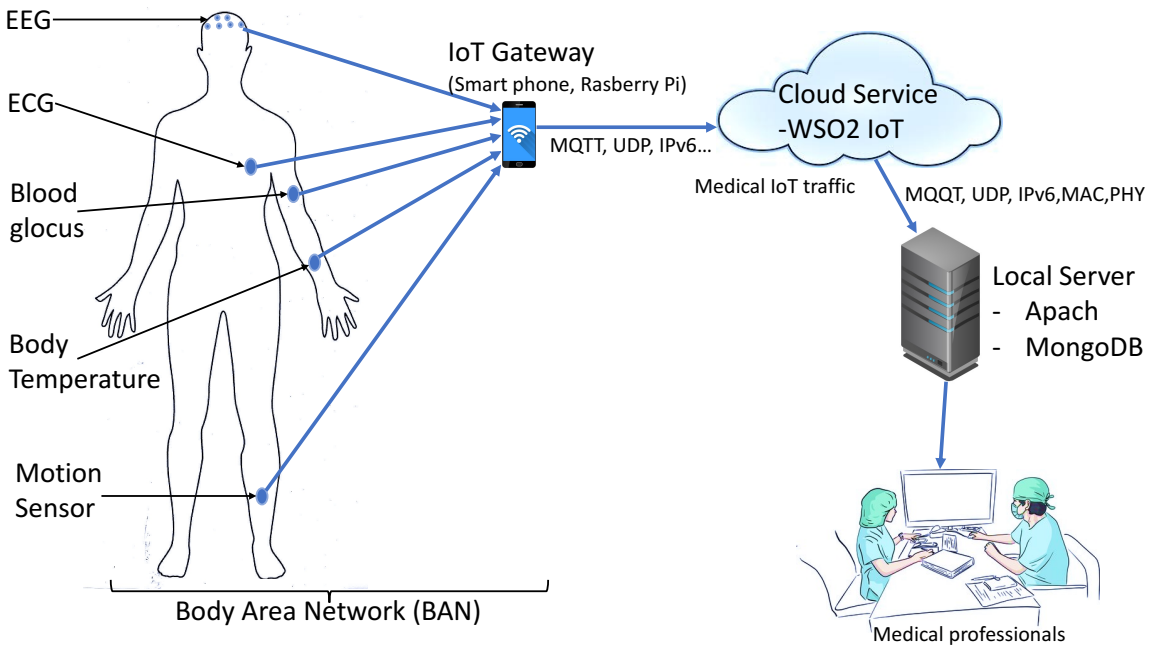


Figure 4.1: Medical IoT network.

The IoT is a communication technology that enables the interconnectivity of all types of devices. A simple architecture of an IoT network for medical devices is shown in Figure 4.1. Like any other communication technology, IoT technology relies on different communication protocols organized in layers to maintain interconnectivity between devices. These protocols include Extensible Messaging and Presence Protocol (XMPP) and Message Queuing Telemetry Transport (MQTT). A comprehensive list of IoT protocols for medical device networks was provided in [26].

Using network technology such as IoT to connect and transfer data from different devices implies that the security requirement of the data depends not only on the devices but also on the network technology. A vulnerability to network protocols and devices puts the generated data in a vulnerable state, and a threat to the network protocols is a threat to the devices and data. In [27], a list of common vulnerabilities of IoT devices and network protocols was proposed, and in [28], a list of common attacks on medical IoT devices and

protocols was presented.

Developing a solution to detect and prevent medical IoT attacks is important in medical cybersecurity, and different approaches have been described in the literature [24] [29]. These approaches can be classified as signature-based, behavior-based, or hybrid. Moreover, the solution logic can be deterministic, stochastic, or both.

From the list of security attacks presented in [28], we focus on the detection of a Distributed Denial of Service (DDoS) attack on medical IoT devices and networks.

DDoS is a security attack on data availability and integrity. It involves an attacker who uses a command and control system to recruit remote devices in a network by inflicting them with malware and later uses these recruited (or zombie) devices to launch an attack simultaneously on a target device, as illustrated in Figure 4.2.

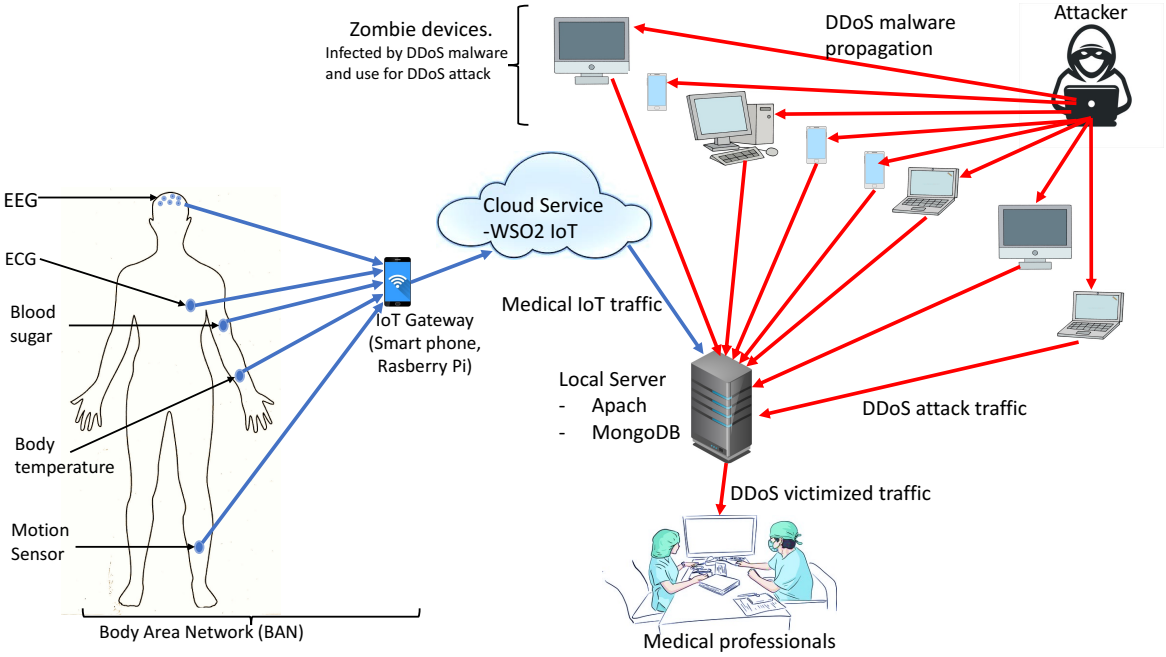


Figure 4.2: DDoS attack operations in an IoT network.

In this study, we focus on both data and IoT device security in medical information technology (IT) infrastructure, using a machine learning approach based on collaborative agents which use heterogeneous data sources based on different input features. Literally, collaboration is the process where two or more entities work together, with the same or different objectives, to complete a task. The importance of a collaborative approach to cybersecurity was presented in [30].

4.2 Model Description

Consider a medical IoT traffic dataset denoted by Q, with each element (i.e., instance) $\{Q_1, Q_2, \ldots\}$ in set Q consisting of two tuples of input and output, expressed as $Q_i = (X_i, y_i)$, where X_i and y_i represent the ith input and a corresponding output instance. The output instance can only be Bagnin or DDoS but not both at the same input instance.

All input instances corresponding to Bagnin output are considered legitimate while those corresponding to DDoS are considered illegitimate. The research problem is to build an agent that can learn to detect if an instance is Bagnin or DDoS. Our proposed solution is depicted in Figure 4.3.

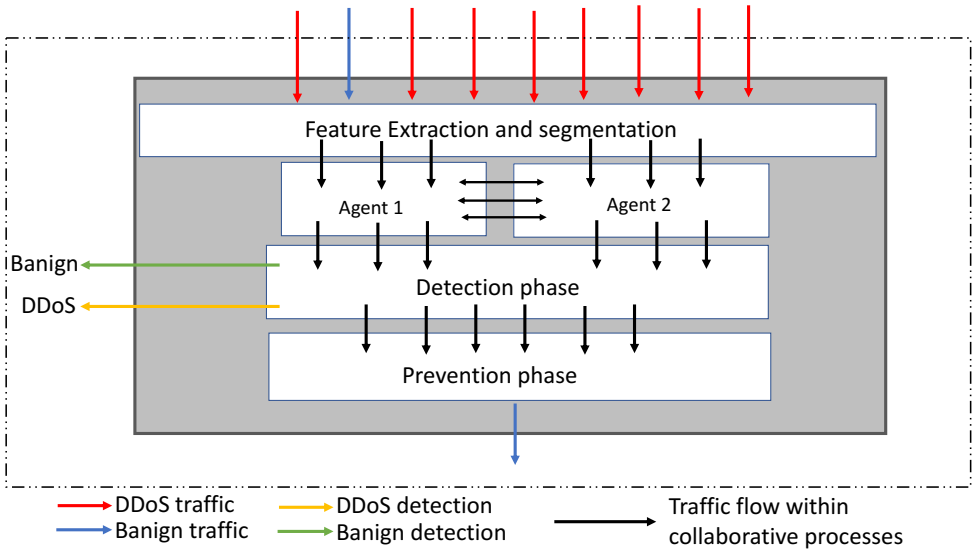


Figure 4.3: DDoS attack detection and prevention operations in an IoT network with the proposed solution.

The proposed solution involves four steps: 1) feature extraction and segmentation of incoming network traffic, 2) agent design, 3) intrusion detection, and 4) intrusion prevention. These steps are illustrated in Figure 4.3. In this study, we focus only on the first three steps. The proposed solution can be installed on a front-end server as a network-based security system for medical IoT edge (i.e., fog) networks.

Feature Extraction and Segmentation

In the feature extraction and segmentation step, each instance of the input properties from traffic data X is extracted and segmented into two (or multiple) sub-vectors X_a and X_b, creating a data subset $Q_a = (X_a, y)$ and $Q_b = (X_b, y)$, respectively. Each sub-vector is considered as the input property of the security target y for a given agent.

The ECU-IoHT medical dataset of five properties was randomly segmented into two

groups, as follows:

$$X = (X_a, X_b) \tag{4.1}$$

where $X_a = (x_1, x_2, x_3)$ and $X_b = (x_4, x_5)$.

The output property y defines the state of the target; in this case, it corresponds to the security state of a medical IoT device. This can include the state of DDoS or other related attacks on the same or different devices.

In the case of the ECU-IoHT medical dataset, we consider y as the true security state of the medical device with regard to a DDoS attack and can take a binary value representing DDoS or Bagnin traffic. Therefore, this output property is dedicated to each agent during the collaborative process.

Agent definition

We consider an agent as an entity that seeks a target. It takes actions (or beliefs) on the target. The actions taken by the agent can be causal or non-causal. In this study, causal action is considered as a conditional action, while non-causal action is considered as a mutual action.

$$P(X_a|y, X_b) = P(X_a|y) \text{and} P(X_b|y, X_a) = P(X_b|y) \tag{4.2}$$

$$\hat{y}_{c,ab} = P(y|X_a, X_b; \theta_{c,a}, \theta_{c,b}, \theta_{m,a}, \theta_{m,b}) \quad = P(y|X_a; \theta_a)P(y|X_b; \theta_{c,b})\left[\frac{P(X_a|X_b; \theta'_b)}{P(X_a)}\right]^{-1} \frac{1}{P(y)}$$

where $P(y|X_a; \theta_a) = \hat{y}_{c,a}$ and $P(y|X_b; \theta_b) = \hat{y}_{c,b}$ are partial causal actions of agents a and b, $\frac{P(X_b|X_a;\theta'_a)}{P(X_b)} = \hat{y}_{m,a}$ and $\frac{P(X_a|X_b;\theta'_b)}{P(X_a)} = \hat{y}_{m,b}$ are mutual actions of agents a and b, and $P(y) = \hat{y}$ is a prior collaborative action of the agents.

The collaborative action $\hat{y}_{c,ab}$ is optimized by simultaneously optimizing the causal and mutual actions of each agent as described in the previous section, and implemented using. In addition, $\hat{y}$ and $\hat{y}_{m,b}$ are respectively used as the regularizer and normalizer of the partial causal predictions.

Furthermore, $\hat{y}$ can also be used to define the global cost of collaboration during learning. This may not be necessary because our approach is based on partial learning to achieve a joint value rather than directly learning the joint value.

The value of n influences the complexity, flexibility, and uncertainty in the collaborative network. The analysis of these properties is reserved as future work.

4.3 Experimental Setup

The dataset for the experiment is based on the ECU-IoHT dataset [31], which has been used for analyzing cyberattacks on the Internet of Health Things [32]. The Intensive Care Unit (ICU) dataset [33] created using the IoT-Flock tool [34] and the ToN_IoT dataset [35] were used to validate the generalization of our model in different environments.

Dataset and Feature Presentation

The ECU-IoHT dataset [31] has eight fields: time stamp of packet, source address of packet, destination address of packet, network protocol of packet, length of packet frame in bytes, information of packet, network status, and attack type.

The type (or network status) field and type of attack field represent the attack labels of the dataset. The network status field was used as the label. Regarding the input features, we excluded the packet information field because it contains packet information reports that may not be helpful in attack detection. Therefore, we used five input features to train the model. Table 4.1 presents statistics of the dataset.

Table 4.1: List of attacks and input fields used in the ECU-IoHT dataset

Attack types	Instances	Input fields used
ARP Spoofing	2359	5
DoS	639	5
Nmap Port scan	6836	5
Smurf Attack	77920	5
No Attack	23454	5
Total instance	111207	

Denial of Service (DoS), and Address Resolution Protocol (ARP).

A DDoS attack constitutes both a DoS attack and port scanning attack. In addition, a Smurf attack is a type of DDoS attack that uses DDoS.smurf malware to render network service inoperable by flooding Internet Control Message Protocol (ICMP) packets to targeted network devices. In this regard, we used DoS, Nmap port scanning, and Smurf attack instances in this experiment.

Therefore, 85395 attack instances and 23454 no-attack instances were used during this experiment, resulting in a total of 108849 attack instances.

Related to the ICU [33] and ToN_IoT [35] datasets, their contents are described in Tables 4.2 and 4.3, respectively.

Table 4.2: List of classes and input fields used in the ICU dataset

Dataset classes	Instances	Input fields used
Attack	80126	50
Environment monitoring	31758	50
Patient monitoring	76810	50
Total instance	188694	

Each class represents a separate csv file which we merge into one file.

Table 4.3: List of attacks and input fields used in ToN_IoT (Network) dataset

Attack types	Instances	Input fields used
Backdoor	20000	43
DDoS	20000	43
DoS	20000	43
Injection	20000	43
Man in the middle(mitm)	1043	43
Password	20000	43
Ransomware	20000	43
Scanning	20000	6
Cross site scripting(xss)	20000	43
Normal	300000	43
Total instance	461043	

The ToN_IoT file used is the Train_Test Network dataset.

Dataset Pre-Processing

The first step in pre-processing is to encode categorical information in the dataset. The label encoding technique was used to encode the target output y, whereas One-Hot encoding was used to encode the source IP, destination IP, and network protocol fields.

Next, normalization operations were performed on the dataset using *minmax scaling*.

$$X_{new} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{4.3}$$

where X is a field in the dataset.

The 108849 attack instances of the dataset were split into training and testing instances, as shown in Table 4.4.

Table 4.4: Partition of attack instances into training and test instances

Attacks	Attack instances	Training instances (70%)	Testing instances (30%)
DoS	639	447	192
Nmap Port scan	6836	4785	2051
Smurf Attack	77920	54544	23376
No Attack	23454	16418	7036
Total instance	108849	76194	32655

The five input fields are divided into two groups. One group consists of the time stamp and the length of the packet fields, whereas the other group consists of the remaining fields. Each collaborative agent was assigned one group of input fields.

We also split the ICU and ToN_IoT datasets into 70% training instances and 30% test instances. One-Hot encoding was used to encode categorical data, while label encoding was used to encode the labels.

Agent Models and Parameters

Table 4.5 presents the simulation parameters of the models used in this experiment.

Table 4.5: Simulation parameters of our model and conventional models

	DNN	Ensemble	Federated	Our model
# of causal net.	1	2	2	2
# of mutual net.	0	0	0	2
# of layers per net.	4	4	4	4, 4
Nodes per layer per causal net.	5,3,2,1	net1:5,3,2,1; net2:5,3,2,1	net1:3,3,2,1; net2:2,3,2,1	net1:3,3,2,1; net2:2,3,2,1
Nodes per layer per mutual net.				net1:3,3,2,1; net2:2,3,2,1
Node type	Sigmoid	Sigmoid	Sigmoid	Sigmoid
Learning algorithm	GD	GD,Bagging	GD,FedAvg	GD,GS
Learning rate	0.61	0.61	0.61	0.61
# of learning cycle	300	300	300	300
Learning value	CE	CE	CE	CE

Deep Neural Network (DNN), Gradient Descent (GD), Gibbs Sampling (GS), Cross Entropy (CE), and # indicates "number".

These parameter settings are for the ECU-IoHT dataset. For the other datasets, only the number of input nodes was changed with respect to their number of features whereas the remaining parameters and hyperparameters were unchanged.

For the ICU dataset, the following number of input nodes per network were used: (50) for DNN, (50, 50) for Ensemble, (25, 25) for Federated, and (25, 25, 25, 25) for our model. For the ToNIoT datasets, the following number of input nodes per network were used: (43) for DNN, (43, 43) for Ensemble, (20, 23) for Federated, and (20, 23, 20, 23) for our model.

Also, the hyper-parameters such as the learning rate, learning cycle (i.e., epoch), the number of layers per network, and the number of nodes per layer, were selected without rigorous hyperparameter turning (optimization) technique. We will investigate hyperparameter training in future work.

4.4 Results and Discussion

ECU-IoHT Dataset Results

The learning and ROC curves for the ECU-IoHT datasets for each model in Table 4.5 are presented in Figure 4.4, 4.5 and 4.6.

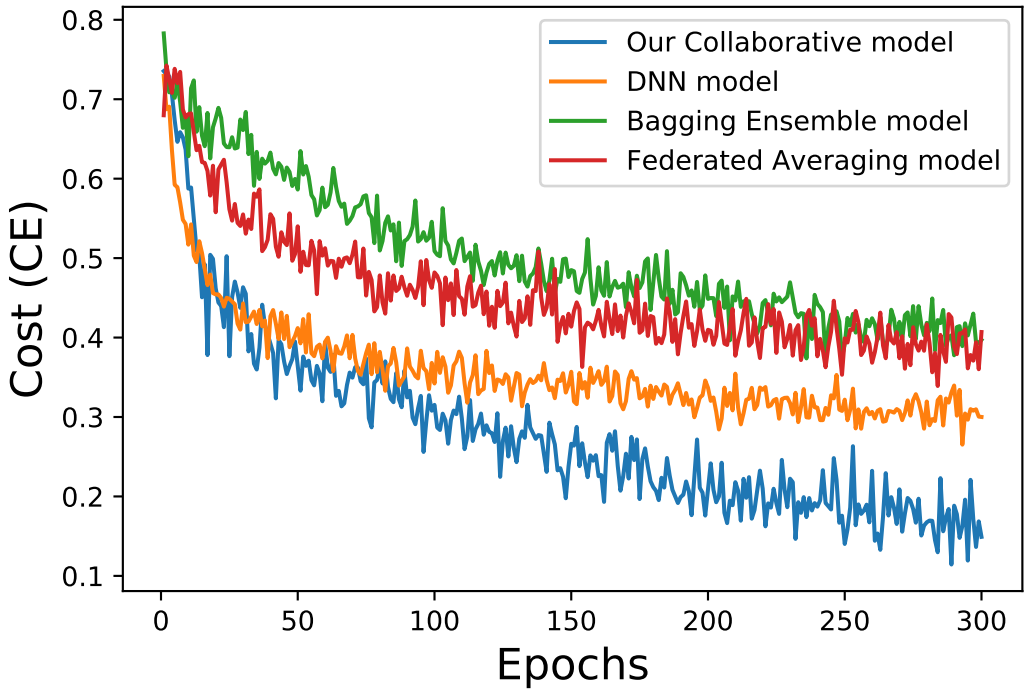


Figure 4.4: Cost-based learning curve of our model on DDoS attack detection compared to other models using ECU-IoHT training dataset.

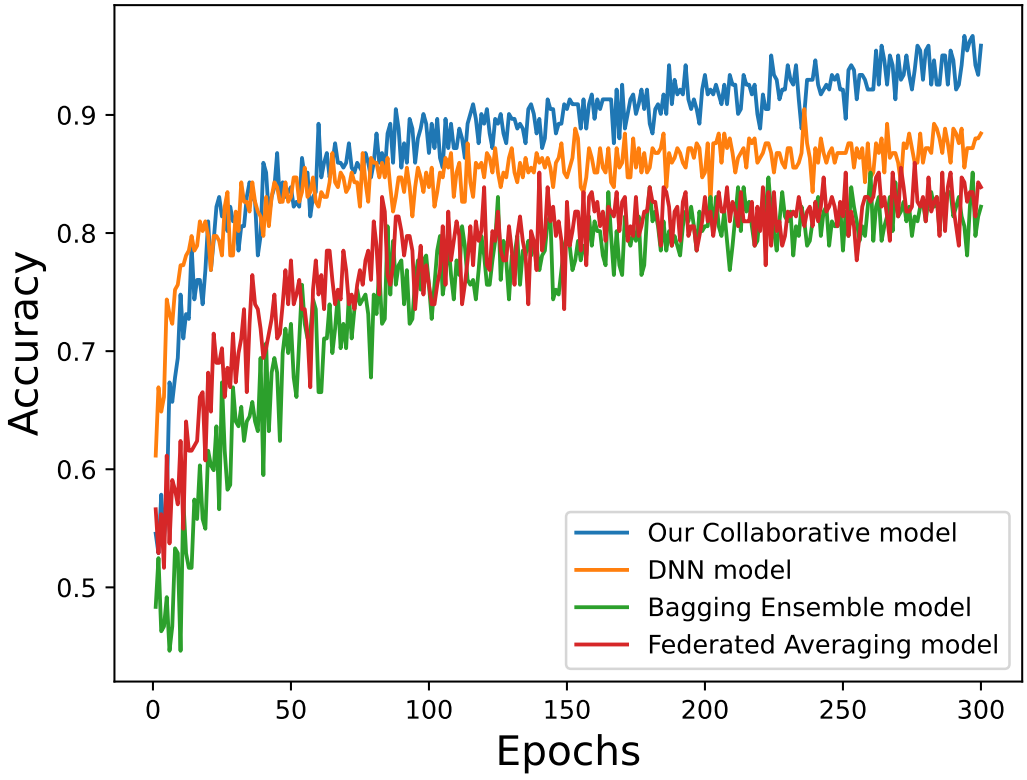


Figure 4.5: Accuracy-based learning curve of our model on DDoS attack detection compared to other models using ECU-IoHT training dataset.

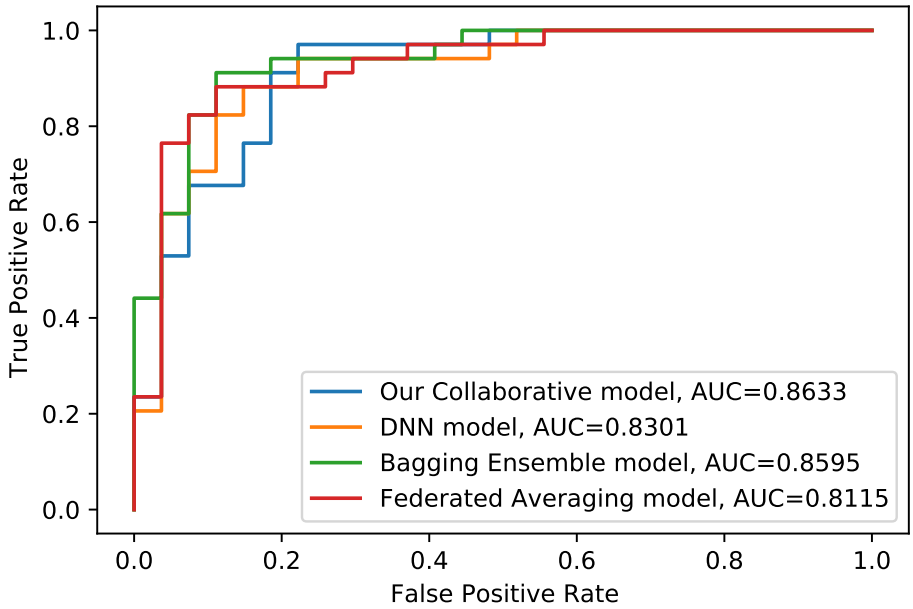


Figure 4.6: ROC curve of our model on DDoS attack detection compared to other models using ECU-IoHT test dataset.

Figure 4.4 illustrates the learning curves of our collaborative agent model compared with the conventional models. From the learning curve, the proposed model reduces the normalized cost value to a lower point of stability than the others. The corresponding accuracies during learning are shown in Figure 4.5. The proposed model turns out to learn more accurately than the other models on the dataset.

However, to confirm the degree of generalization on the dataset, we also performed a prediction performance measurement using the receiver operating characteristic (ROC) function. The results are presented in Figure 4.6, where we observe that our model has a higher AUC ROC than the conventional models and hence can distinguish between target classes more accurately than the other models.

ICU and TON_IoT Datasets Results

The learning and ROC curves obtained by the ICU datasets are shown in Figure 4.7, 4.8 and 4.9, whereas those by the ToN_IoT datasets are shown in Figure 4.10, 4.11 and 4.12, all evaluated for each model in Table 4.5. For both datasets, we observe that our model has a higher AUC, accuracy, and a lower normalized cost value than the conventional models and generalizes on the dataset better than the other models.

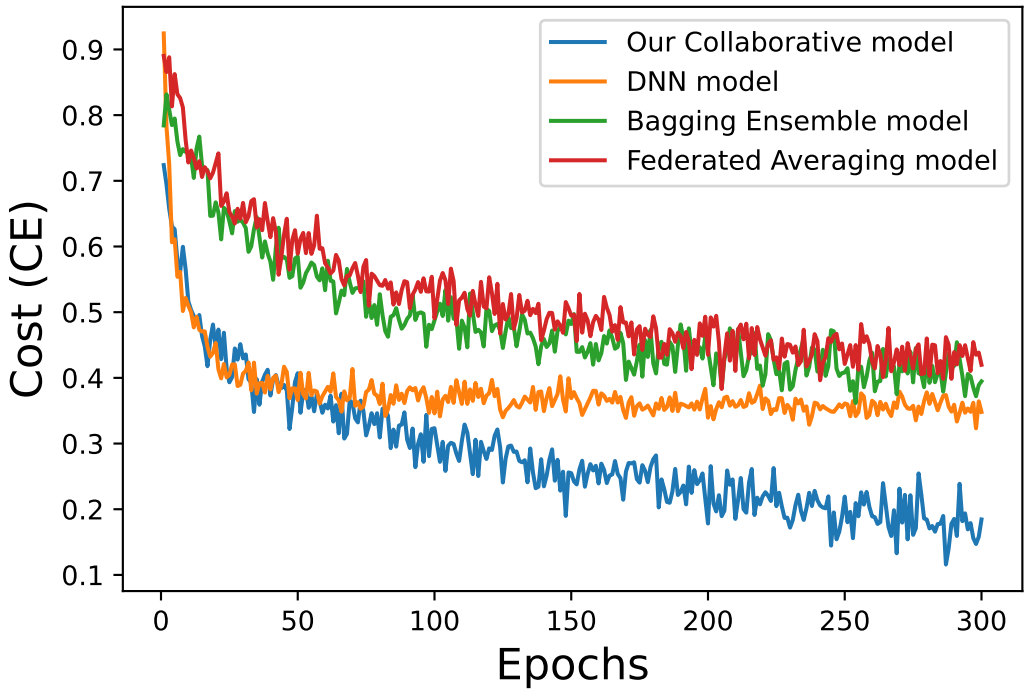


Figure 4.7: Cost-based learning curve of our model on DDoS attack detection compared to other models using ICU training dataset.

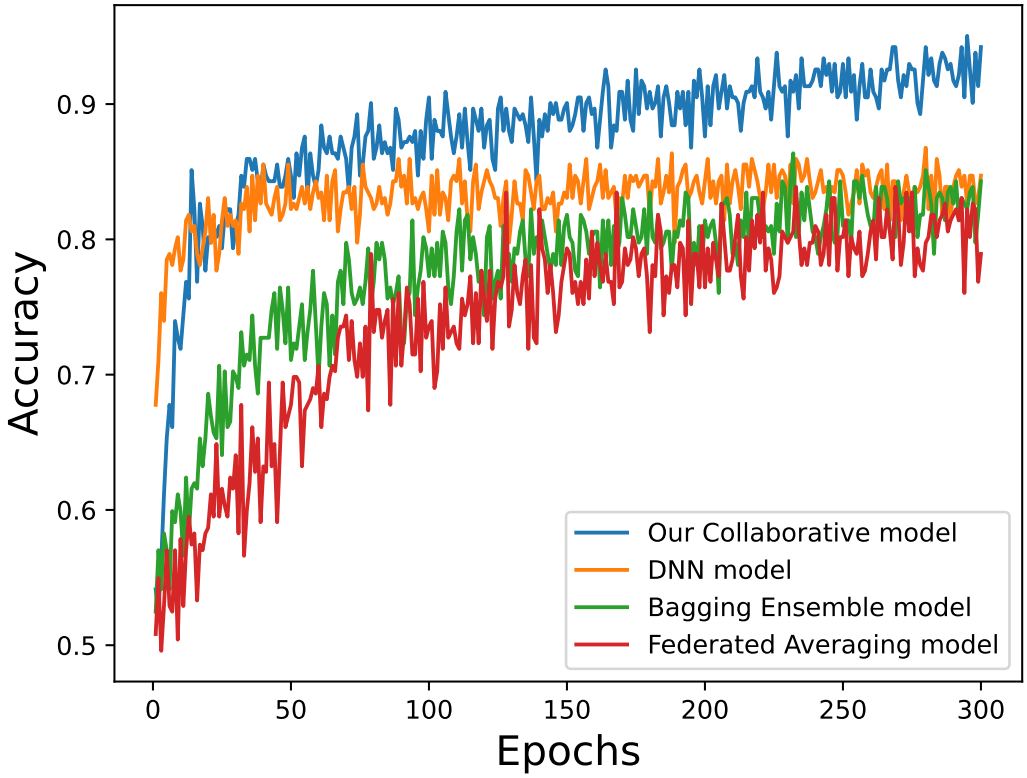


Figure 4.8: Accuracy-based learning curve of our model on DDoS attack detection compared to other models using ICU training dataset.

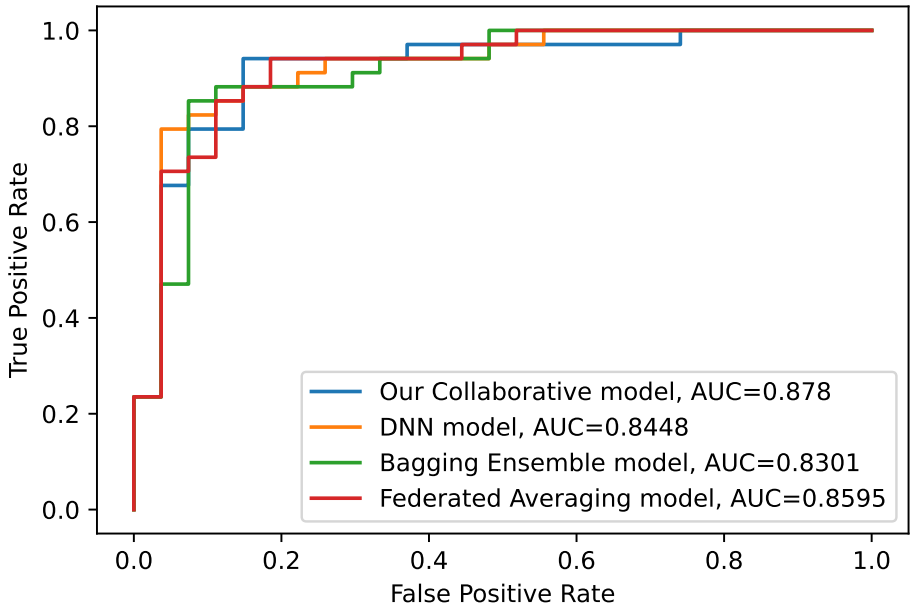


Figure 4.9: ROC curve of our model on DDoS attack detection compared to other models using ICU test dataset.

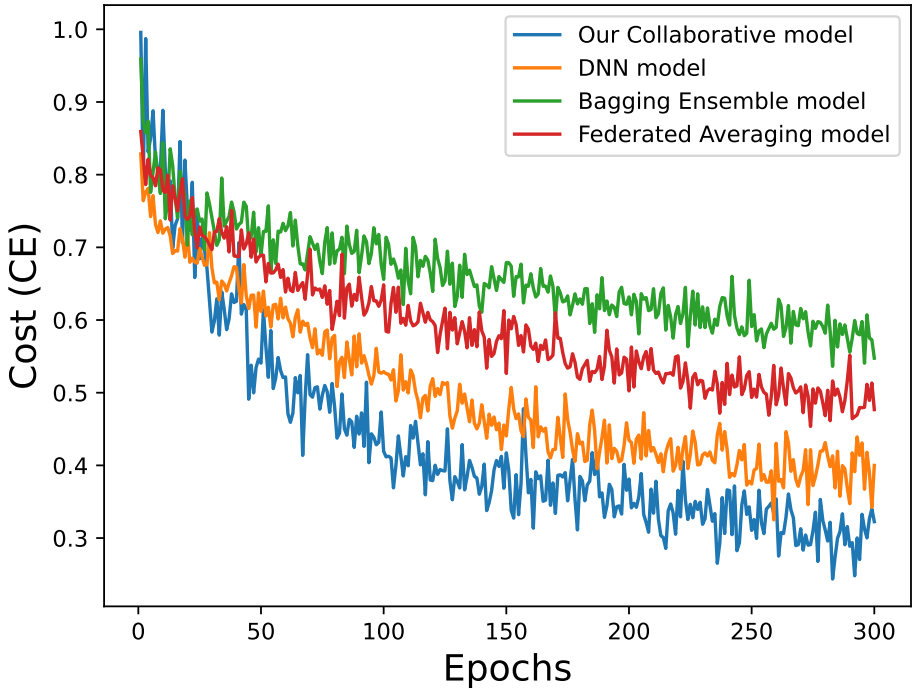


Figure 4.10: Cost-based learning curve of our model on DDoS attack detection compared to other models using ToN_IoT training dataset.

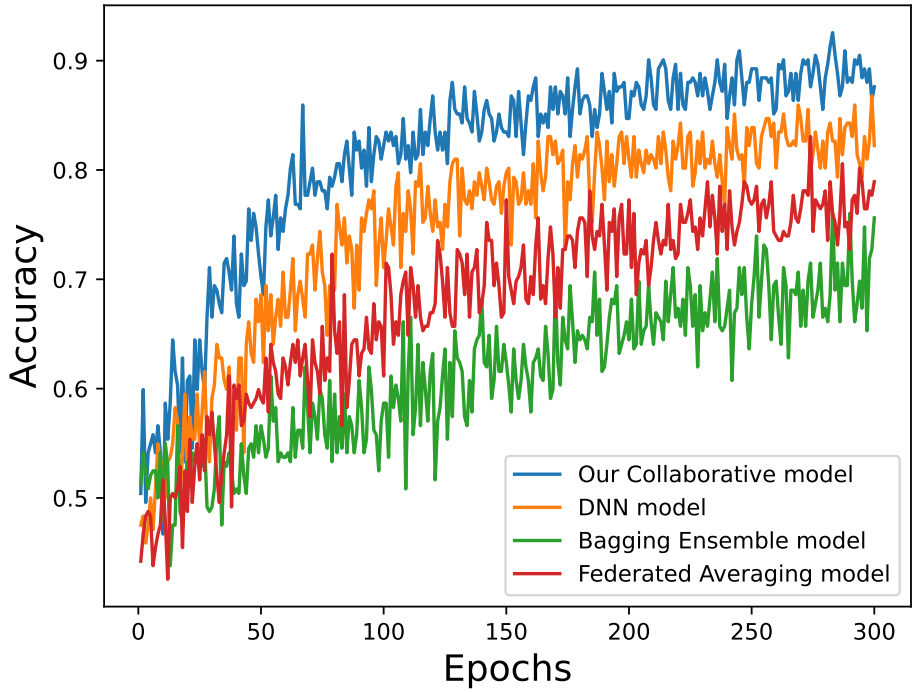


Figure 4.11: Accuracy-based earning curve of our model on DDoS attack detection compared to other models using ToN_IoT training dataset.

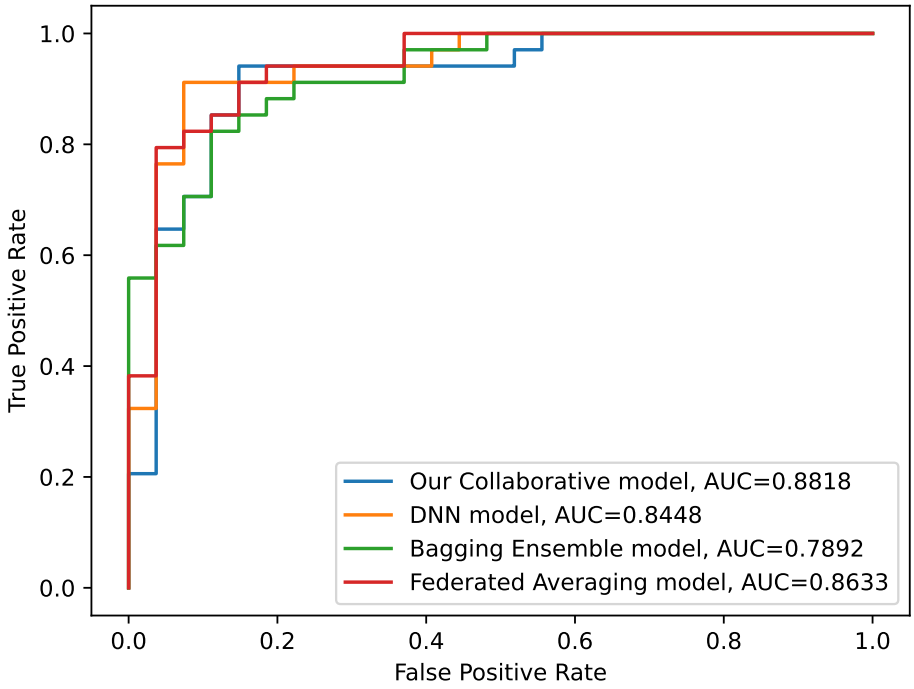


Figure 4.12: ROC curve of our model on DDoS attack detection compared to other models using ToN_IoT test dataset.

Numerical Comparisons

Table 4.6: Predictive performance and comparison of the models

Datasets	Models	Acc(%)	Prec(%)	Rec(%)	F-s (%)	△AIC
ECU-IoHT	DNN	98.17	95.10	89.04	91.21	36.83
ECU-IoHT	Ensemble	98.78	93.03	96.08	94.26	74.54
ECU-IoHT	Federated	97.59	94.10	90.74	92.36	65.16
ECU-IoHT	Our model	**99.99**	**99.99**	**99.86**	**99.57**	**110.78**
ICU	DNN	99.19	93.73	86.91	96.55	37.14
ICU	Ensemble	99.48	96.89	91.99	97.93	75.32
ICU	Federated	99.01	96.32	90.28	95.59	66.07
ICU	Our model	**99.96**	**99.97**	**99.87**	**99.64**	**109.02**
ToN_IoT	DNN	98.80	94.91	90.89	93.90	37.10
ToN_IoT	Ensemble	98.99	98.19	94.57	99.99	72.11
ToN_IoT	Federated	98.92	95.81	92.91	97.80	63.52
ToN_IoT	Our model	**99.98**	**99.95**	**99.65**	**99.97**	**112.78**

Accuracy (Acc), Precision (Prec), Recall (Rec), F-score (F-s), and Akaike Information Criterion delta score (△AIC) with $AIC_{min} = 0$.

Table 4.6 compares the models in terms of their accuracy, precision, recall, F-score, and Akaike Information Criterion delta score (△AIC) on the prediction of the test datasets. It demonstrates that our model performs well under most of the prediction metrics on all the datasets. Due to its large number of parameters, it turns out to have a higher △AIC value than the other models. However, such a large number of parameters is due to the collaborative framework, which is more of a merit than a demerit.

Apart from these predictive (statistical) performance results, we also evaluated the computational performance on the prediction actions of the models based on the floating-point operations (FLOPs), floating-point operations per second (FLOPS), and multiply accumulate operations (MACs) computational complexity measures. The experiments were performed on a quadcore 4GHz processor with 16 double-precision (DP) FLOPs per cycle (i.e., 256 GFLOPS DP), but the required FLOPS of the models were estimated using the latency (execution time) and FLOPs of their predictions.

Furthermore, the uncertainties (epistemic) of the models for each dataset were also evaluated using the ensemble method with 25 ensembles for each model.

Table 4.7: Computational performance and uncertainty of the models

Datasets	Models	FLOPs	Latency	FLOPS	MACs	ME	2σ-U
ECU-IoHT	DNN	1.1M	21ms	51.90M	0.55M	94.09	0.053
ECU-IoHT	Ensemble	2.2M	23ms	94.74M	1.1M	101.12	0.081
ECU-IoHT	Federated	1M	30ms	33.31M	0.5M	105.69	0.094
ECU-IoHT	Our model	**2M**	**27ms**	**74.12M**	**1M**	**69.27**	**0.032**
ICU	DNN	18.86M	38ms	0.53G	9.43M	112.55	0.074
ICU	Ensemble	37.72M	41ms	0.90G	18.86M	120.93	0.052
ICU	Federated	18.86M	45ms	0.41G	9.43M	118.01	0.071
ICU	Our model	**37.75M**	**42ms**	**0.94G**	**18.88M**	**70.04**	**0.023**
ToN_IoT	DNN	29.24M	56ms	0.54G	14.62M	93.90	0.094
ToN_IoT	Ensemble	58.48M	57ms	1.02G	29.24G	134.90	0.063
ToN_IoT	Federated	29.24M	61ms	0.52G	14.62M	125.14	0.081
ToN_IoT	Our model	**58.51M**	**59ms**	**1.18G**	**29.26M**	**89.04**	**0.051**

Floating-point operations (FLOPs), Floating-point operation per second (FLOPS), Multiply accumulate operations (MACs), median error (ME), 2σ uncertainty (2σ-U), Mega (M), millisecond (ms), Giga (G).

As shown in Table 4.7, the FLOPs, latency, FLOPS, and MACs values of our model are higher than the other models. This is due to its collaborative framework consisting of many computing agents. However, the generalization (i.e., error) and generalization uncertainty (i.e., error variability) of our model is better than that of the other models on all the datasets because of its lower median and 2σ uncertainty values, respectively. This can also be confirmed by the graphs of the normalized cost values in Section 4.2.1–4.2.2.

Also, the fact that the required FLOPS of the models are less than the 256 GFLOPS can

be due to many factors such as computational overload, memory operations, and latency degradation. For example, the high latency of the federated agents is due to an additional latency caused by the communication between the local and central agents.

Other performance measures of interest that are beyond the scope of this study are the stability, robustness, throughput, memory operations, and energy usage of the models. These are reserved for future work.

Lastly, in Table 5.3 we compared the statistical performance results of our model with those of other studies using the same datasets in medical IoT security.

Table 4.8: Comparison of our model with models from other research works

References	Models	Datasets	Acc(%)	Prec(%)	Rec(%)	F-s(%)
Ilhan et al. [36]	MLP	ECU-IoHT	99.99	99.99	99.9	99.99
Current work	Collabo	ECU-IoHT	**99.99**	**99.99**	**99.86**	**99.57**
Ilhan et al. [36]	MLP	ICU	99.94	99.94	99.92	99.93
Hussain et al. [33]	NB	ICU	52.18	79.67	99.70	68.50
	KNN	ICU	99.48	99.65	99.68	99.58
	RF	ICU	99.51	99.70	99.79	99.65
	AB	ICU	99.50	99.55	99.44	99.47
	LR	ICU	99.50	95.28	90.35	94.70
	DT	ICU	99.47	99.69	99.79	99.63
Current work	Collabo	ICU	**99.96**	**99.97**	**99.87**	**99.64**
Ilhan et al. [36]	MLP	ToN_IoT	98.12	98.15	98.12	98.12
Khan et al. [37]	XSRU	ToN_IoT	99.38	99.39	98.99	99.37
Zachos et al. [38]	DT	ToN_IoT	99.97	99.97	99.91	99.94
	NB	ToN_IoT	34.44	27.91	99.97	43.64
	LR	ToN_IoT	98.70	95.52	99.55	97.50
	RF	ToN_IoT	99.96	99.89	99.95	99.92
	KNN	ToN_IoT	99.98	99.98	99.97	99.96
	SVM	ToN_IoT	98.73	95.30	99.93	97.56
Current work	Collabo	ToN_IoT	**99.98**	**99.95**	**99.65**	**99.97**

Multi-Layer Perceptron (MLP), Naive Bayes (NB), K-Nearest Neighbor (KNN), Random Forest (RF), AdaBoost (AB), Logistic Regression (LR), Decision Tree (DT), Explainable simple recurrent units Internet of Medical Things (XSRU-IoMT), and Support Vector Machine (SVM)

From the prediction results in Table 5.3, our model outperforms most of the other models on the ICU and ToN_IoT datasets under most of the metrics, but the model proposed by Ilhan et al. [36] has a better recall and F-score on the ECU-IoHT dataset and a better recall on the ICU dataset. In addition, the DT and KNN models proposed by Zachos et al. [38] have better precision and recall. These are some of the few instances in which the other models outperformed the proposed model.

Chapter 5
Application 2: Text Classification in NLP using GenCO

5.1 Background

Textual communication has always been one of the predominant methods of communication in human society since the invention of writing by different cultures around the world. Added to the increase in human interactions in recent years, a large amount of textual information is generated daily from different sources, such as web pages, blogs, emails, social media, and articles.

To understand the content of textual information, many language analysis tasks, such as lexical (or morphological) analysis, syntax analysis (or parsing), semantic analysis, topic modeling, and text classification, can be performed on the text corpus. In this paper, we focus on text classification and provide a machine-learning solution to automate the classification of textual information.

Text classification consists of assigning a sentence or document to an appropriate predefined category . This category can involve topic, sentiment, language, or all. So, text classification tasks may include news classification, emotion classification, sentiment analysis, citation intent classification, spam classification, and so on. This is important in many institutions for creating archives and the organization of large amounts of text needed for effective planning and decision-making on their projects.

In general, depending on the type of category, text classification can be divided into topic classification, sentiment classification (or analysis), language classification, and hybrid classification based on any two or all three of these categories.

The pipeline of a general text classification task in NLP is presented in [39]. The initial step in this pipeline is text preprocessing, where the text corpus is processed for case harmonization, noise removal, tokenization, stemming, lemmatization, normalization, feature extraction, and vectorization. After preprocessing, the vectorized features are then fed into a classification algorithm that outputs a prediction of the category which defines

the given text corpus.

During preprocessing, case harmonization involves setting all text in the corpus to the same case, that is, either lowercase or uppercase. Noise removal involves the removal of stop words and special characters from the corpus. Tokenization involves splitting the text into small chunks of words or sentences, called tokens. Stemming involves reducing words to their root or base form. Lemmatization involves breaking a word down to its root meaning to identify similarities.

Furthermore, normalization is the mapping of different, but semantically equivalent, phrases onto a single canonical representative phrase. It can be divided into morphological normalization, such as stemming and lemmatization, lexical normalization, such as spelling correction, syntactic normalization, such as grammatic correction, and semantic normalization, such as synonyms elimination. Since stemming and lemmatization are types of morphological normalization techniques, the normalization step usually involves lexical, syntactic, and/or semantic normalization.

Feature extraction involves the selection of features required for training and classification. Vectorization involves converting selected features from a text corpus into numerical vectors. Different types of vectorization techniques are used in NLP, such as Bag-of-Words, Term Frequency–Inverse Document Frequency (TF-IDF), Word2Vec, and Continuous Bag-of-Words (CBOW) vectorizations.

The importance of text classification has led to the development of many algorithms to automate the process . Such algorithms may use either a deterministic, stochastic, or hybrid approach to infer the category of a text corpus. The advancement in machine learning algorithms as a stochastic logical operation has increased the use of the stochastic approach in text classification.

Machine learning algorithms can broadly be divided into symbolic (mostly rule-based), statistical (mostly data-driven and discriminative), and probabilistic (mostly generative) models. Each of these models can be further divided into supervised, semi-supervised, unsupervised, self-supervised, and reinforcement learning. Furthermore, based on their number of input and output variables, they can be classified as single-input single-output (SISO), multiple-input single-output (MISO), single-input multiple-output (SIMO), and multiple-input multiple-output (MIMO) models. Multiple output (MO) models are also called multitarget models, which include multilabel models for classification tasks. In this paper, we use a MISO-based generative supervised learning model for text classification.

A major challenge in text classification algorithms is the classification of a heterogeneous text corpus (or corpora) . Heterogeneous text corpuses (or corpora) are those with latent relationships between their features (i.e., vocabularies), and their classification is challenging. The vocabularies of a text (i.e., a document) in a corpus or a corpus in corpora discussing housing prizes and sports will be explicitly unrelated, but the understanding of any implicit (or hidden) relationship between their vocabularies can help in their classifi-

cation. The degree of such a heterogeneous relationship between features may vary and will influence the classification of a text corpus (or corpora).

This degree of heterogeneity between features of the same corpus or different corpora is different from the similarity measure between documents, which is estimated using cosine similarity and hamming distance measures, for example. The heterogeneity measure used in this study focuses on capturing the correlation in terms of the probabilistic dissimilarity between features in the same corpus or different corpora, while conventional similarity measures focus on capturing similarity between documents in the same corpus or different corpora.

Furthermore, the number of vectorized features in most text classification tasks is very large such that conventional text classification models cannot perform very well due to the curse of dimensionality. A common technique to solve this problem is to use an n-gram language model, where $n > 1$. The downside of this technique is that it leads to a sparsed vectorization, which is less useful in generating a reliable classification result, especially when n increases. Also, many preprocessing operations, such as defining the count limits of feature occurrences, are performed to reduce unwanted features.

These are the challenges we seek to solve in this paper, and we do so using a collaborative learning model that takes into account the relationship between the features of a text corpus and their dimensionality.

In this study, we propose a generative model based on collaborative partial classifications as a solution to the problem of a heterogeneous text corpus and the curse of dimensionality in text classification. We performed experiments to evaluate the performance of our model on different heterogeneous text datasets and compared the outcome with results from other studies. We propose a method to explain the classification results of our proposed model.

5.2 Model Description

In this section, the conventional and proposed models related to text classification in this study are presented.

Conventional model

The conventional approach to classifying text makes use of all the extracted features of the text corpus to predict the given category. This can be represented mathematically as

$$\hat{y} \triangleq f(X) \tag{5.1}$$

where X is a vector of the text features, $\hat{y}$ is the predicted value of the text category, and $f(.)$ is a function defining the prediction process.

Different conventional models are used to implement this prediction process. These include Naive Bayes (NB), Support Vector Machine (SVM), Deep Neural Networks (DNN), Bidirectional Encoder Representations from Transformers (BERT), and Recurrent Neural Networks (RNN). Figure 5.1 represents the conventional text classification using NB.

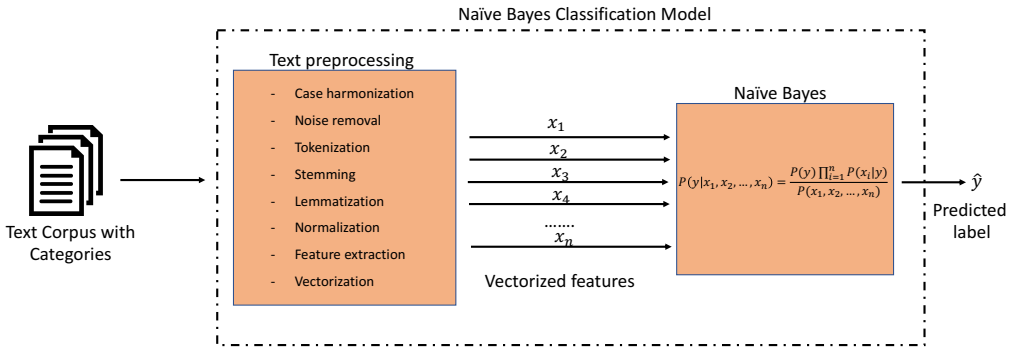


Figure 5.1: Text classification using Naive Bayes.

Considering a news corpus with a feature vector $X = (x_1, x_2, \ldots, x_n)$ and category set $y = \{y_1, y_2, \ldots, y_m\}$, where the elements of y are mutually exclusive. Using a conventional probabilistic learning model such as NB, the text classification can be expressed based on the Bayes rule as follows:

$$\hat{y} = P(y|x_1, x_2, \ldots, x_n) = \frac{P(y)P(x_1, x_2, \ldots, x_n|y)}{P(x_1, x_2, \ldots, x_n)} \tag{5.2}$$

where n is the number of features in X, $P(y)$ is the prior distribution of the category y, $\hat{y}$ is the posterior distribution of the category y, $P(X|y)$ is the likelihood of the category y given all text features in X (i.e., probability of X given y), and $P(x_1, x_2, \ldots, x_n) = P(X)$ is the marginal distribution of the text features (also called evidence).

The final predicted category (or class) $\hat{y}_k$ of y is defined by the class y_k with the maximum probability over all categories.

$$\hat{y}_k = \underset{k\in\{1,2,\ldots,m\}}{\arg\max}\left(\frac{P(y_k)P(x_1, x_2, \ldots, x_n|y_k)}{P(x_1, x_2, \ldots, x_n)}\right) \tag{5.3}$$

where m is the number of categories in y, $y = \{y_1, y_2, \ldots, y_m\}$, and $P(y_k)$ is the probability of a given category y_k of y.

Given that $P(x_1, x_2, \ldots, x_n)$ is independent on y, then

$$\hat{y}_k \propto \underset{k\in\{1,2,\ldots,m\}}{\arg\max}\ (P(y_k)P(x_1, x_2, \ldots, x_n|y_k)) \tag{5.4}$$

In this way, $P(x_1, x_2, \ldots, x_n)$ is considered as a normalizing factor that depends only on X, and, thus, will be a constant if the values of all the features of X are known.

The learning process based on this Bayesian inference model is then defined as an update operation that aims to maximize the predicted distribution $\hat{y}$ over the cumulative instances of X and y.

$$\max(\hat{y}) = \underset{j\in\{1,2,\ldots,l\}}{\arg\max}\left(\frac{P(y^{(j)})P(X^{(j)}|y^{(j)})}{P(X^{(j)})}\right) \tag{5.5}$$

where j is the numbering of the cumulative instances (also considered here as the learning or update time), and l is the total instances of X and y, i.e., the data size.

As l increases, the probability improves, but increase in l will also increase the computational complexity of the learning and inference process. Thus, this model is preferable with a small data size. Nevertheless, unlike data-hungry models, such as deep neural networks that require a large data size to perform well, this Bayesian model does perform well with small data sizes.

From this Bayesian inference and learning, NB is defined using the assumption of mutual independence between the features of X conditioned on the category y.

$$P(x_i|X, y) = P(x_i|y) \tag{5.6}$$

Thus, NB inference and learning models can be obtained from the Bayesian model as follows,

$$\hat{y}_k = \underset{k\in\{1,2,\ldots,m\}}{\arg\max}\left(\frac{P(y_k)\prod_{i=1}^{n}P(x_i|y_k)}{P(x_1, x_2, \ldots, x_n)}\right) \tag{5.7}$$

$$\max(\hat{y}) = \underset{j\in\{1,2,\ldots,l\}}{\arg\max}\left(\frac{P(y^{(j)})\prod_{i=1}^{n}P(x_i^{(j)}|y^{(j)})}{P(X^{(j)})}\right) \tag{5.8}$$

As an example, consider the sentences from a news corpus:

1. The weather is worse today due to climate change.
2. The increase in economic crises is due to the pandemic.
3. World leaders are determined to end world crises.
4. Major decisions to end climate change were made by world leaders at the climate summit.
5. During the pandemic, economic activities were shut down, making world leaders struggle with the world economy.

6. No world economy survives the pandemic.

7. World climate change summit discusses how to tackle world climate change crises during a pandemic crisis.

8. Most world leaders don't have a large economy to tackle the pandemic and climate change crises.

9. Without a sustainable economy, it may take longer to survive the pandemic shock.

10. World economic crises and pandemics are headaches to world leaders.

After preprocessing the news corpus, consider that the extracted vectorized features and labels are those shown in Table 5.1. The task is to classify each sentence into Business (B) or Geography (G) news based on the extracted features.

Table 5.1: Bag-of-Words (i.e., 1-gram word) vectorization of a news corpus.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	y
1	0	0	0	1	1	0	0	G
2	1	1	1	0	0	0	0	B
3	0	1	1	0	0	2	1	B
4	0	0	0	2	1	1	1	G
5	2	0	1	0	0	2	1	B
6	1	0	1	0	0	1	0	B
7	0	2	1	2	2	2	0	G
8	1	1	1	1	1	1	1	G
9	1	0	1	0	0	0	0	B
10	1	1	1	0	0	1	1	B

x_1 denotes economy, x_2 denotes crises, x_3 denotes pandemic, x_4 denotes climate, x_5 denotes change, x_6 denotes world, x_7 denotes leader, y denotes class.

Using NB, we first compute the prior of each class, then calculate the likelihoods, and finally estimate the posterior by multiplying the prior with the likelihood since the marginal is fixed. A Laplacian smoothing is used as a normalizer to avoid the probability of zero. A log probability is used to avoid computational underflow (i.e., floating point underflow) in the case of many features.

Computing the prior and the likelihood is given as follows

$$\text{Class priors: } P(c) = \frac{n_c}{n} \tag{5.9}$$

$$\text{Likelihoods: } P(w|c) = \frac{n_{w,c} + a}{n_c + a|V|}, \quad a = 1 \tag{5.10}$$

where c is a class, n_c is the frequency (number of occurrences) of a class, n is the total occurrences of all the classes, w is a feature, $n_{w,c}$ is the frequency (number of occurrences) of a feature given a class c, a is a smoothing parameter which is equal to 1 for Laplacian smoothing, and V is the number of vectorized features.

There are many variants of NB but the two most popular variants used for text classification are Multinomial NB and Bernoulli NB. For example, using a Bernoulli NB model on Table 5.1 will require Table 5.1 be transformed to a binary Bag-of-Words vectorization, as shown in Table 5.2; then, Equations (5.9) and (5.10) will be applied to Table 5.2 for prior and likelihood estimations, respectively.

Table 5.2: Binary Bag-of-Words (i.e., 1-gram word) vectorization of a news corpus.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	y
1	0	0	0	1	1	0	0	G
2	1	1	1	0	0	0	0	B
3	0	1	1	0	0	1	1	B
4	0	0	0	1	1	1	1	G
5	1	0	1	0	0	1	1	B
6	1	0	1	0	0	1	0	B
7	0	1	1	1	1	1	0	G
8	1	1	1	1	1	1	1	G
9	1	0	1	0	0	0	0	B
10	1	1	1	0	0	1	1	B

x_1 denotes economy, x_2 denotes crises, x_3 denotes pandemic, x_4 denotes climate, x_5 denotes change, x_6 denotes world, x_7 denotes leader, y denotes class.

Using Equation (5.9), the prior estimations with respect to Table 5.2 will be

$$P(G) = 4/10, P(B) = 6/10$$

Using Equation (5.10), the likelihood estimations will be

$$P(x_1|G) = \frac{1+1}{4+7} = \frac{2}{11}, P(x_1|B) = \frac{5+1}{6+7} = \frac{6}{13}$$

$$P(x_2|G) = \frac{2+1}{4+7} = \frac{3}{11}, P(x_2|B) = \frac{3+1}{6+7} = \frac{4}{13}$$

$$P(x_3|G) = \frac{2+1}{4+7} = \frac{3}{11}, P(x_3|B) = \frac{6+1}{6+7} = \frac{7}{13}$$

$$P(x_4|G) = \frac{6+1}{4+7} = \frac{7}{11}, P(x_4|B) = \frac{0+1}{6+7} = \frac{1}{13}$$

$$P(x_5|G) = \frac{4+1}{4+7} = \frac{5}{11}, P(x_5|B) = \frac{0+1}{6+7} = \frac{1}{13}$$

$$P(x_6|G) = \frac{3+1}{4+7} = \frac{4}{11}, P(x_6|B) = \frac{4+1}{6+7} = \frac{5}{13}$$

$$P(x_7|G) = \frac{2+1}{4+7} = \frac{3}{11}, P(x_7|B) = \frac{3+1}{6+7} = \frac{4}{13}$$

For the same text classification problem, the Bernoulli model will be,

$$P(G|x_1, x_2, x_7) \propto \frac{2}{11} \times \frac{3}{11} \times \frac{7}{13} \times \frac{1}{13} \times \frac{1}{13} \times \frac{5}{13} \times \frac{3}{11} \times \frac{4}{10} = 0.00017$$

$$P(B|x_1, x_2, x_7) \propto \frac{6}{13} \times \frac{4}{13} \times \frac{3}{11} \times \frac{7}{11} \times \frac{5}{11} \times \frac{4}{11} \times \frac{4}{13} \times \frac{6}{10} = 0.00075$$

Since the probability for the statement to be Business news is higher than that to be Geography news, the sentence is classified as Business news.

As shown in the prediction and learning processes of conventional Bayesian learning models, such as NB, no consideration is made to capture the relationship between the features. This is considered to be computationally expensive, especially when estimating the marginal $P(X)$ for large data size and number of features. This leads to approximate solutions, such as (5.4), that ignore such complexities.

As previously explained, the elimination of such a relationship will affect the classification performance of the model, and eliminate the possibility to manage the heterogeneity between the features of the text corpus. In the next section, we propose an inference and learning model that takes into account such a relationship and applies it to classify heterogeneous text corpora.

Proposed model

Unlike the conventional approach that makes use of all the features of the text corpus to infer and learn the given category, our proposed approach provides the possibility to segment the input features into multiple groups of input features, forming different corpora, on which separate learning and inference are performed. This model is defined mathematically using probabilistic logic as follows:

$$P(X_i|X, y) = P(X_i|y) \tag{5.11}$$

$$\hat{y} = P(y|X) = \frac{1}{P(y)^{n-1}} \prod_{i=0}^{n} P(y|X_i) \left(\frac{P(X_{i+1}| \bigcap_{\mu=0}^{i} X_\mu)}{P(X_{i+1})} \right)^{-1} \tag{5.12}$$

where y is the given set of categories, X is a vector of feature vectors $X = (X_0, X_1, \ldots, X_n)$ and $X_i = (x_1, x_2, \ldots)$, $\hat{y}$ is the posterior distribution (i.e., class posterior) of y given X, $P(y)$ is the prior distribution (i.e., class prior) of y based on X, $P(y|X_i)$ is the partial posterior distribution (i.e., partial class posterior) of y given X_i, $P(X_{i+1})$ is the prior distribution (i.e., observation prior) of X_{i+1} based on $\bigcap_{\mu=0}^{i} X_\mu$, $P(X_i| \bigcap_{\mu=0}^{i} X_\mu)$ is the posterior distribution (i.e., observation posterior) of X_i given $\bigcap_{\mu=0}^{i} X_\mu$, and n is the number of features vectors.

It is worth noting that the posterior probability distributions can be interpreted as likelihood functions, i.e., the posterior of y given X is similar to the likelihood of X given y, and so on. Thus, the term posterior is used interchangeably with the term likelihood in this manuscript.

The inference and learning based on this proposed model can then be expressed as

$$\begin{aligned}
\hat{y}_k &= \underset{k \in \{1,2,\ldots,m\}}{\arg\max} P(y_k|X) \\
&= H(X) \underset{k \in \{1,2,\ldots,m\}}{\arg\max} \left(\frac{1}{P(y_k)^{n-1}} \prod_{i=0}^{n} P(y_k|X_i) \right) && (5.13) \\
\Rightarrow \hat{y}_k &\propto \underset{k \in \{1,2,\ldots,m\}}{\arg\max} \left(\frac{1}{P(y_k)^{n-1}} \prod_{i=0}^{n} P(y_k|X_i) \right) && (5.14) \\
\max(\hat{y}) &= \underset{j \in \{1,2,\ldots,l\}}{\arg\max} P(y_k^{(j)}|X^{(j)}) && (5.15)
\end{aligned}$$

where $H(X) = \prod_{i=0}^{n} \left(\frac{P(X_{i+1}| \bigcap_{\mu=0}^{i} X_\mu)}{P(X_{i+1})} \right)^{-1}, \quad H(X) \in [0, \infty]$

During inference and learning, estimating the priors and likelihoods (i.e., posteriors) based on this model for both the class and observation is given as

$$\text{Class prior: } P(c) = \frac{n_c}{n} \tag{5.16}$$

$$\text{Observation prior: } P(w) = \frac{n_w}{n} \tag{5.17}$$

$$\text{Class Likelihood: } P(c|w) = \frac{n_{c,w} + a}{n_w + a|V|}, \quad a = 1 \tag{5.18}$$

$$\text{Observation Likelihood: } P(w_i|w_j) = \frac{n_{w_i,w_j} + a}{n_{w_j} + a|V|}, \quad a = 1 \tag{5.19}$$

where n is the total number of instances in the text vectorization, c is a class, n_c is the

frequency (number of) occurrences of a class, w is a feature vector, n_w is the frequency of occurrences of a feature vector w, $n_{c,w}$ is the frequency of occurrence of a class c given the occurrence of a feature vector w, n_{w_i,w_j} is the frequency of occurrence of a feature vector w_i given the occurrence of another feature vector w_j, a is a smoothing parameter, which is equal to 1 for Laplacian smoothing, and V is the number of vectorized features.

Using this proposed model, for any given classification task, the first step is to segment the features into parts, forming separate feature vectors, then inference and learning are applied on each of the feature vectors using the class likelihood (or partial class posterior) $P(c|w)$. The partial class posterior can also be expressed as an inference problem on each segmented feature where our model or a Bayesian model can be applied to generate its results. The results from each partial class posterior are then integrated (or aggregated in the case of a logarithmic scale) to represent the classification result on the whole text corpus.

Segmenting a feature vector into sub-vectors while maintaining the relationship between the features can be a daunting task. One way to approach this is to organize the segmented feature vectors in a sequence, then to apply Proposition 5.2, taking into account the heterogeneity between the features in the sequence. This heterogeneity between the features is given by the value of $H(X)$ and is independent of y.

As an example, using the vectorized features defined in Table 5.2 with one feature per segment, the following priors and likelihoods can be estimated.

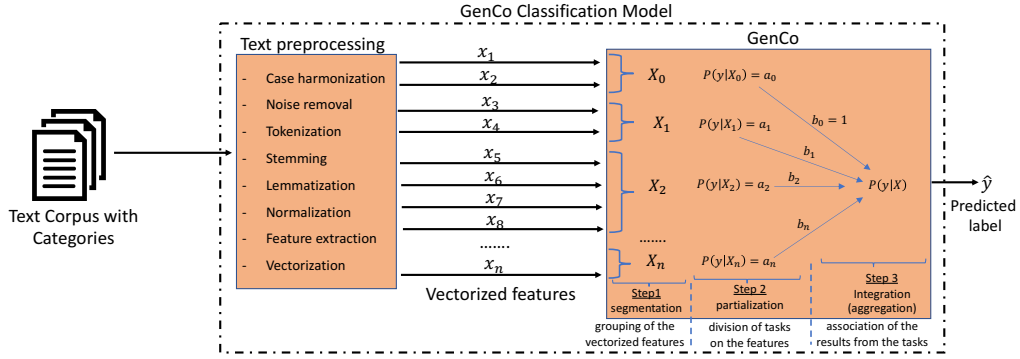


Figure 5.2: Text classification using GenCo.

Using Equation (5.16), the class priors will be

$$P(G) = 4/10, P(B) = 6/10$$

Using Equation (5.17), the observation priors will be

$$P(x_1 = yes) = 6/10, P(x_2 = yes) = 5/10,$$

$$P(x_3 = yes) = 8/10, P(x_4 = yes) = 4/10,$$

$$P(x_5 = yes) = 4/10, P(x_6 = yes) = 7/10,$$

$$P(x_7 = yes) = 5/10$$

Using Equation (5.18), the class likelihoods will be

$$P(G|x_1) = \frac{0+1}{6+7} = \frac{1}{13}, P(G|\neg x_1) = \frac{3+1}{4+7} = \frac{5}{11},$$

$$P(G|x_2) = \frac{2+1}{5+7} = \frac{3}{12}, P(G|\neg x_2) = \frac{2+1}{5+7} = \frac{3}{12},$$

$$P(G|x_3) = \frac{2+1}{8+7} = \frac{3}{15}, P(G|\neg x_3) = \frac{2+1}{2+7} = \frac{3}{9},$$

$$P(G|x_4) = \frac{4+1}{4+7} = \frac{5}{11}, P(G|\neg x_4) = \frac{0+1}{6+7} = \frac{1}{13},$$

$$P(G|x_5) = \frac{4+1}{4+7} = \frac{5}{11}, P(G|\neg x_5) = \frac{0+1}{6+7} = \frac{1}{13},$$

$$P(G|x_6) = \frac{3+1}{7+7} = \frac{4}{14}, P(G|\neg x_6) = \frac{1+1}{3+7} = \frac{2}{10},$$

$$P(G|x_7) = \frac{2+1}{5+7} = \frac{3}{12}, P(G|\neg x_7) = \frac{2+1}{5+7} = \frac{3}{12},$$

Using Equation (5.19) and ordering the features to be conditionally dependent from x_1 to x_7 with a Markov property of a level 1 assumption, the observation likelihoods will be

$$P(x_2|x_1) = \frac{3+1}{6+7} = \frac{4}{13}, P(x_2|\neg x_1) = \frac{1+1}{4+7} = \frac{2}{11}$$

$$P(x_3|x_2) = \frac{5+1}{5+7} = \frac{6}{13}, P(x_3|\neg x_2) = \frac{0+1}{5+7} = \frac{1}{13}$$

$$P(x_4|x_3) = \frac{1+1}{8+7} = \frac{2}{15}, P(x_4|\neg x_3) = \frac{2+1}{2+7} = \frac{3}{9}$$

$$P(x_5|x_4) = \frac{4+1}{4+7} = \frac{5}{11}, P(x_5|\neg x_4) = \frac{0+1}{6+7} = \frac{1}{13}$$

$$P(x_6|x_5) = \frac{3+1}{4+7} = \frac{4}{11}, P(x_6|\neg x_5) = \frac{4+1}{6+7} = \frac{5}{13}$$

$$P(x_7|x_6) = \frac{5+1}{7+7} = \frac{6}{14}, P(x_7|\neg x_6) = \frac{0+1}{3+7} = \frac{1}{10}$$

This implies that, using the Bernoulli version of our proposed model, the classification of a corpus given that it has the words economy, crises and leaders, will be,

$$P(G|x_1, x_2, x_7) \propto \left(\frac{4}{10}\right)^{-6} \times \frac{1}{13} \times \frac{3}{12} \times \frac{3}{9} \times \frac{1}{13} \times \frac{1}{13} \times \frac{2}{10} \times \frac{3}{12} = 0.012$$

$$P(B|x_1, x_2, x_7) \propto \left(\frac{6}{10}\right)^{-6} \times \frac{7}{13} \times \frac{4}{12} \times \frac{1}{9} \times \frac{7}{13} \times \frac{7}{13} \times \frac{3}{10} \times \frac{4}{12} = 0.046$$

Similar to the classification results using conventional models, our proposed model also classifies the statement as Business news but with larger floating point value than the conventional model. Furthermore, the heterogeneity between the features at the last prediction instance can be estimated using the heterogeneity function $H(X)$ as follows:

$$H(X) = \prod_{i=1}^{7} \left(\frac{P(x_{i+1}| \bigcap_{\mu=1}^{i} x_\mu)}{P(x_{i+1})} \right)^{-1} = \frac{\frac{4}{13}}{\frac{5}{10}} \times \frac{\frac{1}{13}}{\frac{8}{10}} \times \frac{\frac{1}{9}}{\frac{6}{10}} \times \frac{\frac{7}{13}}{\frac{6}{10}} \times \frac{\frac{3}{13}}{\frac{3}{10}} \times \frac{\frac{1}{10}}{\frac{5}{10}} = 0.001513$$

We consider the reciprocal of this value as a form of mutuality between the features, which measures their similarity (i.e., homogeneity), and can be used to indirectly measure their heterogeneity in this model. Thus, the mutuality $M(X)$ between the features will be,

$$M(X) = \frac{1}{H(X)}, \quad M(X) \in [0, \infty] \tag{5.20}$$

Therefore, if $H(X) = 0.001513$, then $M(X) = 660.983143$. This implies that there is more probabilistic similarity than dissimilarity between the features. In other words, the features are probabilistically more joined together than disjoint in their occurrence.

In this study, $M(X)$ measures the correlation (or association) between the features in terms of the probabilistic similarity (i.e., homogeneity) of their dependency on one another. For any two features X_i and X_j, where each one is conditioned on the other, if $M(X_i, X_j) = 1$, then $P(X_i|X_j) = P(X_i)$ and $P(X_j|X_i) = P(X_j)$, which implies X_i and X_j are probabilistically identical, but non-similar in their dependence to each other. If $M(X) > 1$, then $P(X_i|X_j) > P(X_i)$ and $P(X_j|X_i) > P(X_j)$, which implies X_i and X_j have a direct (i.e., increase in value when conditioned) probabilistic similarity in their dependency to one another. If $M(X) < 1$, then $P(X_i|X_j) < P(X_i)$ and $P(X_j|X_i) < P(X_j)$, which implies X_i and X_j have an indirect (i.e., decrease in value when conditioned) probabilistic similarity in their dependency to one another. Using M(X) in the logarithmic scale will lead to $M(X) = 0$

(region of no mutuality), $M(X) > 0$ (region of increasing mutuality), and $M(X) < 0$ (region of decreasing mutuality), respectively. These also apply to the dissimilarity measure $H(X)$.

$M(X)$ can be compared with conventional similarity measures in NLP, such as the cosine similarity and hamming distance measures. However, unlike conventional similarity measures which are separated from the classification model, our proposed similarity measure $M(X)$ and dissimilarity measure $H(X)$ form part of our classification model; hence, they can be used to explicitly explain the classification results.

One advantage of this proposed model in text classification rests on the fact that it enables the breakdown of a high computational classification problem into smaller less computational classification problems. This may be better in terms of conventional models, whose computational complexity increases with the number of features due to the computation of the marginal distribution. The Markov property can also be applied to reduce such complexity in both the conventional and our proposed models.

Also, the use of a heterogeneity function or homogeneity function in the model, rather than a marginal function as in conventional probabilistic models, enables clear visibility of the influence of the relationship between the features to the learning and prediction processes of the model. We shall present a mutuality matrix to capture the probabilistic variation in the homogeneous relationship between the features during learning and show how this variation influences the learning and prediction results of the model.

Furthermore, this model can be expressed as an algebraic series as follows:

$$f(y|X) = a_0 b_0 \times a_1 b_1 \times a_2 b_2 \times a_3 b_3 \times \ldots \times a_n b_n \tag{5.21}$$

where $a_0 = P(y|X_0)$, $a_1 = P(y|X_1)$, $\ldots$, $a_n = P(y|X_n)$, $b_0 = 1$, $b_1 = \frac{H(X_1)}{P(y)}$, $b_2 = \frac{H(X_2)}{P(y)}$, $\ldots$, $b_n = \frac{H(X_n)}{P(y)}$, $H(X_1) = \frac{P(X_1)}{P(X_1|X_0)}$, $H(X_2) = \frac{P(X_2)}{P(X_2|X_0,x_1)}$, $\ldots$, $H(X_n) = \frac{P(X_n)}{P(X_n|\bigcap_{\mu=0}^{n-1} X_\mu}$), and $X = (X_0, X_1, X_2, \ldots, X_n)$.

The first term a_0 is considered as a bias partial class posterior in the model, and b is a combination of the heterogeneity (or mutuality) and regularization values. In general, $P(y)$ is considered to act as a regularizer to each heterogeneous (or mutual) relationship $H(X_i)$, while $H(X_i)$ acts as a normalizer of the partial class posterior a_i. In this way, $H(X)$ acts as a normalizer of the merged (integrated or aggregated) class posterior $f(y|X)$, while $(P(y))^{1-n}$ acts as a regularizer of $H(X)$.

This representation transforms the model into a network of partial actions, as shown in Figure 5.2. Such a representation is useful for mathematical analysis and the network can be expanded to multiple segmentation and partialization layers; however, this will increase its design and computational complexities. This network is different from conventional Bayesian networks (i.e., belief networks) and Markov networks (i.e., Markov random field), which focus on using the marginal distribution of the input features and ignore the

heterogeneity between the input features.

To avoid computational overload, the conditional expressions between the features can be reduced to fewer dependent features through the application of the Markov property. Also, each term in the series can be normalized using logarithmic normalization to avoid computational underflow.

5.3 Experimental Setup

Two important aspects of the experiments are the model definition and the datasets.

Dataset and Feature Presentation

The datasets (i.e., text corpora) used for the experiment included the Twitter US Airline Sentiment dataset [40] for sentimental analysis, the Conference Paper dataset [41] for topic classification, and the SMS Spam dataset [42] for spam classification, as shown in Table 5.3.

Table 5.3: Statistics of the datasets.

Datasets	Documents	Vocabularies	Vocabulary Segments	Categories
(1) Twitter US Airline dataset [40]	14,640	100	10	3
(2) Conference Paper dataset [41]	2507	100	10	5
(3) SMS Spam dataset [42]	5574	50	5	2

Dataset Pre-Processing

The general data preprocessing steps, as shown in Figure 5.2, were used for all the datasets as part of the text classification pipeline. These data preprocessing steps were performed using the Python NLTK library. The vectorized vocabulary for each dataset was generated using Python sklearn CountVectorizer. A lower and upper bound frequency of the vocabularies was set to reduce non-semantic vocabularies and computational complexity.

Each vectorized dataset was later split into 70% training and 30% test instances. Label encoding was used to encode the labels.

Model Definition

The model used during this experiment is based on Proposition 5.2 and is defined by the following hyperparameters:

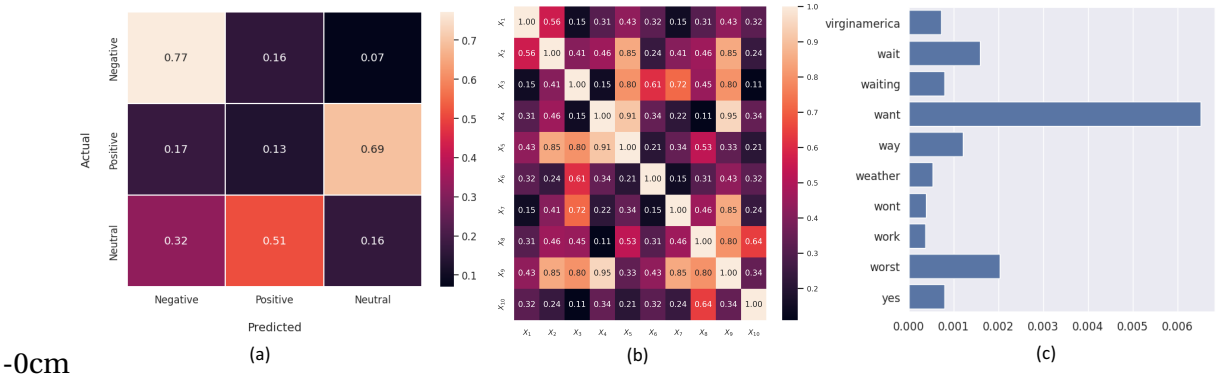
- Smoothing parameter: The smoothing parameter $a \in (0, 1]$, which is fixed to $a = 1$, corresponding to a Laplacian smoothing.
- Number of segmentation: The number of segments used depends on the number of features (i.e., dimensions of the vocabulary) of the dataset concerned.
- Number of partialization layers: A single layer of partialization is used, and the number of partial class posteriors in the layer is equal to the number of feature segments.

5.4 Results and Discussion

The classification results for the experiments are presented in this section, together with the homogeneity measure between the features of each dataset using a mutuality matrix. Also, the confusion matrix and classification report based on each dataset is presented. Lastly, we provide a comparison of the classification results of our model for each dataset with models from different studies that used the same dataset.

Twitter US Airline Dataset Results

The confusion matrix of our model based on the Twitter US Airline dataset is presented in Figure 5.3, together with the mutuality matrix of the 10 feature segments.



-0cm

Figure 5.3: Results with the Twitter US Airline dataset. (**a**) Confusion matrix. (**b**) Mutuality matrix for the 10 feature segments. (**c**) Combined mutuality of each feature in the segment with the highest mutuality.

The results presented in Figure 5.3a show that the proposed model classifies negative labels better than both positive and neutral labels. To explain this behavior of the model, we generate the mutuality matrix for the 10 feature segments, as presented in Figure 5.3b.

The mutuality between every two feature segments is defined using (5.20), applying a level 1 Markov property assumption and Axiom 6.2.1. This results in the diagonal values next to the leading diagonal values in Figure 5.3b. The mutuality matrix at this level gives information about the interrelationship between the feature segments. As shown in Figure 5.3b, most of these relationships are decreasing mutual relationships because, for every two feature segments, X_i and X_j, $M(X_i, X_j) < 1$.

We further examined the mutuality of the features in each segment and found that the segment X_{10} with the highest combined mutuality contained features with semantically negative sentiments and whose combined mutuality was amongst the highest in all the segments, such as the words "worst" and "wait" in Figure 5.3c. This implies that mutuality (and heterogeneity) between the features or feature segments plays an essential role in the classification process of this model; hence, they can be used to explain its classification results.

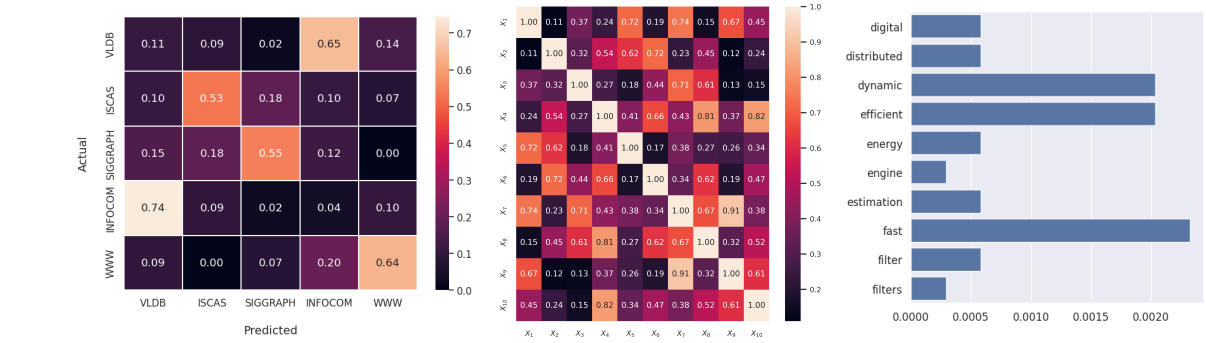
The type of semantic distinctions (classification) of features with respect to a class label during label classification is considered in this study to represent the semantic intelligence of the model on the labels, i.e., the ability to understand the meanings of the labels. This implies that training the predictive (causal) intelligence of this model will imply training its semantic intelligence and vice versa. However, one should not expect the semantic logic of the model on the text to always be similar to human semantic logic applied to the same text, since the model may use a different semantic logic from humans, although it is formally trained for human semantic awareness.

Conference Paper Dataset Results

The confusion matrix of our model based on the Conference Paper dataset is presented in Figure 5.4, together with the mutuality matrix of the 10 feature segments.

The results presented in Figure 5.4a show that the model classifies the WWW label better than the other labels. Using the mutuality matrix defined for the 10 feature segments as presented in Figure 5.4b, we also obtain information about the interrelationships between the feature segments, where segment X_4 has the highest combined mutuality.

Looking further into the mutuality of the features in each segment, we also found that the segment X_4 with the highest combined mutuality contained features which were semantically related to the WWW label and whose combined mutuality was amongst the highest in all segments, such as the words "dynamic", "efficient" and "fast", as shown in Figure 5.4c. Nevertheless, the low normalized true positive (TP) value for the WWW label implies some features in the label were not semantically classified by the model under the WWW label. The incorrect semantic classification of features with respect to the labels using mutual value allocation may account for the low TP of the other labels.

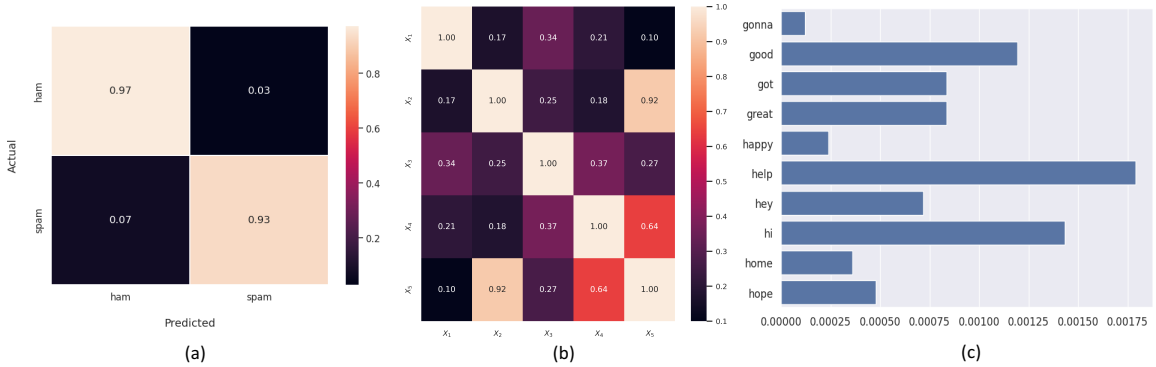


-0cm

Figure 5.4: Results with the Conference Paper dataset. (**a**) Confusion matrix. (**b**) Mutuality matrix for the 10 feature segments. (**c**) Combined mutuality of each feature in the segment with the lowest mutuality.

SMS Spam Dataset Results

The confusion matrix of our model based on the SMS Spam dataset is presented in Figure 5.5, together with the mutuality matrix of the five feature segments.



-0cm

Figure 5.5: Results with SMS Spam dataset. (**a**) Confusion matrix. (**b**) Mutuality matrix for the 5 feature segments. (**c**) Combined mutuality of each feature in the segment with the lowest mutuality.

The results in Figure 5.5a show that the model classifies the “ham” label better than the “spam” label. Using the mutuality matrix defined for the five feature segments as presented in Figure 5.5b, segment X_2 has the highest combined mutuality; the combined mutuality of each of its features is presented in Figure 5.5c. The features with high com-

bined mutuality include words such as “help” and “hi”, which are common words in spam SMS, while words such as “good”, “got” and “hi” are common words used in ham SMS. The high mutual value allocation on both ham and spam words explains the high true positive (TP) results for both ham and spam labels in Figure 5.5a.

Chapter 6
Application 3: Target Tracking in ADAS/AD Radar using CoPreMo

6.1 Background

Autonomous driving (AD) vehicles are vehicles that are capable of driving and parking without the need for human assistance. This is achieved through the use of technologies such as Advanced driver assistance system (ADAS) technologies, which can be divided into sensing and control technologies.

It is predicted in [43] that by 2030, many cars will be fully autonomous. This will eliminate many traffic accidents caused by human factors, such as tiredness, drunkenness, fading memory, poor sight, and so on. However, such innovations cannot be reliable if their performance is not well optimized, because this may instead lead to more accidents.

Data from the World Health Organization (WHO) shows that about 1.35 million people die each year from road accidents, mostly by car, during a rear-end collision, angle collision, head-on collision, and side-sweep collision. This poses a challenge to ADAS/AD vehicles because for them to be reliable, they must have technologies that can enable them to overcome such challenges.

To detect the different collisions that may occur to an ADAS/AD vehicle, radars are placed at different locations on the ADAS/AD vehicle. The location of different long and short-range radars on an ADAS/AD vehicle is presented in Figure 6.1. Corner radars (rear and front) are mostly short-range and are used for blind-spot detection, lane-change assist, and front/rear cross-traffic alert. Front radars are used for autonomous emergency braking and adaptive cruise control, while rear radars are used for parking assistance.

Technically, ADAS technologies can be divided into sensing and control technologies. Sensing technologies are those that are used to collect information about the environment in which the ADAS/AD vehicle is found. They include cameras, radars, lidars, sonars, and so on. Control technologies are those that are used to control the vehicle in the environment. They include a model predictive controller (MPC), proportional–integral–derivative

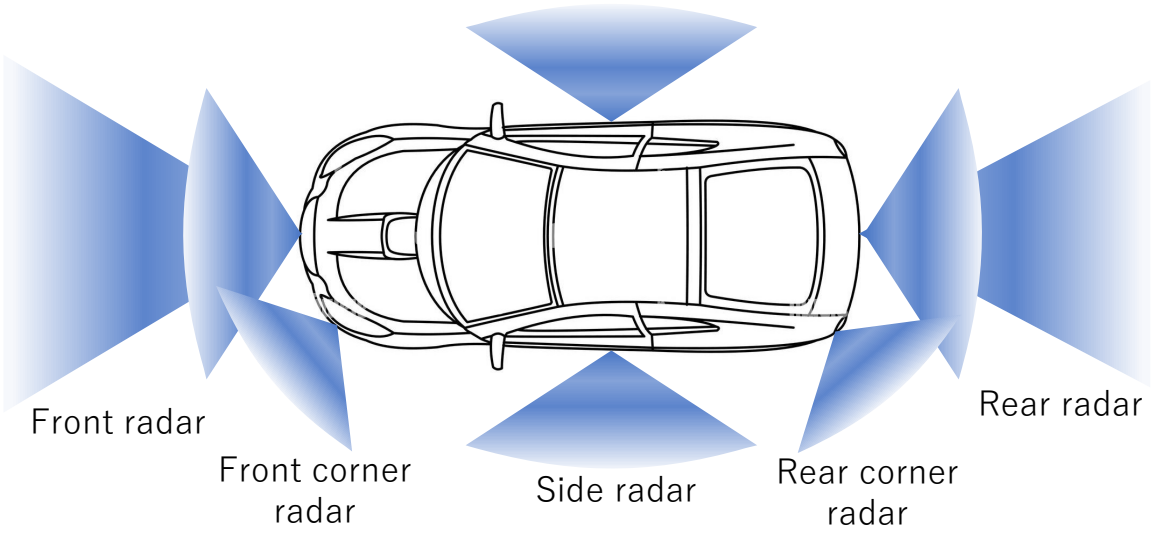


Figure 6.1: ADAS vehicle with sensors location based on traffic scenario.

(PID) controller, and so on.

In this study, we focus on tracking technologies, specifically radar-based tracking. Actually, a radar is a device that uses radio waves to detect the range (distance), speed, and angle of any object in an environment. Detecting the presence of objects, their range, speed, and angle in a traffic environment are important functions in ADAS/AD vehicles that can limit accidents such as traffic collisions.

The reliability of these radar functions depends on their performance in terms of their limits, resolution, and accuracy. In this study, we focus on the performance of the radar range estimation function. This implies the range limits, resolution, and accuracy of the radar. The reliability in terms of the durability of the radar and the robustness of its measurements are not discussed in this study.

The general architecture of a radar system together with its algorithm is presented in Figure 6.2. As shown in the figure, the radar is the sensory part of an ADAS/AD system. Signals from the radar antenna enter the ranging and tracking units by passing through a series of steps. This signal is then used by the control unit to enable the ADAS/AD vehicle to take action on a target.

An important function of ADAS/AD vehicles is the tracking of detected targets. Once a target has been detected by a radar or any other sensors, these targets need to be tracked over time. Tracking is an estimation process in the time domain. Most of the sensors estimate their properties using the feature space of their input and do not have functions to estimate the next direction of the target they have detected. To carry out this function,

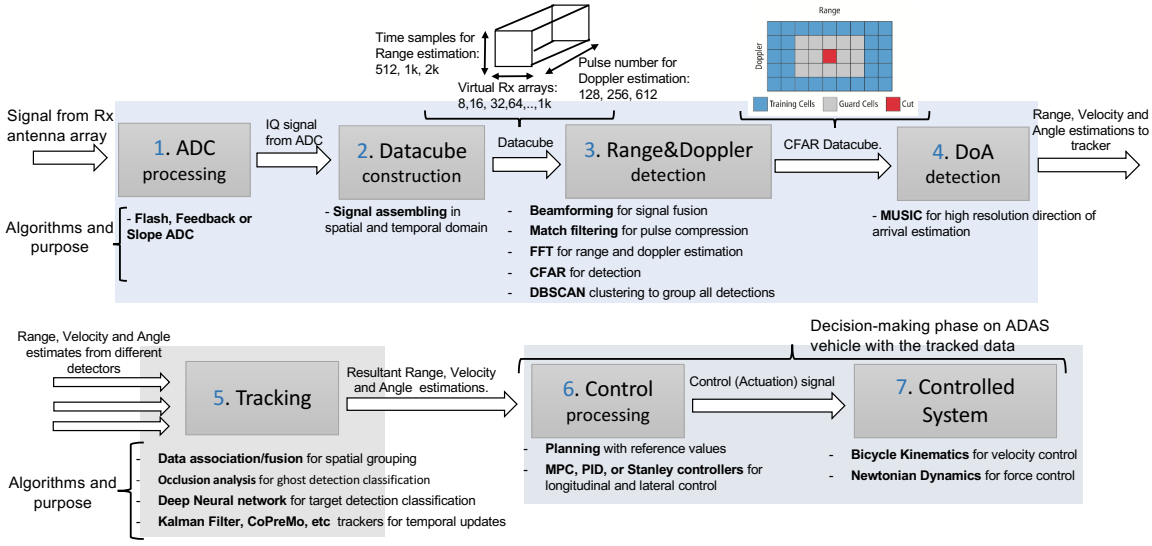


Figure 6.2: General architecture for a radar system and control unit.

tracking actions are used.

In this study, we proposed a tracking model for radar targets based on the CoPreMo model, a collaborative predictive model in time series as explained in Chapter 3. The tracking model is described in the next section.

6.2 Model Description

The collaborative framework is defined as follows

Axiom 6.2.1 (Conditional independence of input properties)

$$P(X_i|X,y) = P(X_i|y) \tag{6.1}$$

Proposition 6.2.1

$$\hat{y} = P(y|X) = \frac{1}{P(y)^{n-1}} \prod_{i=0}^{n} P(y|X_i) \left(\frac{P(X_{i+1}| \bigcap_{\mu=0}^{i} X_\mu)}{P(X_{i+1})} \right)^{-1} \tag{6.2}$$

where y is the given set of categories, X is a vector of feature vectors $X = (X_0, X_1, \ldots, X_n)$ and $X_i = (x_1, x_2, \ldots)$, $\hat{y}$ is the posterior distribution (i.e., class posterior) of y given X, $P(y)$ is the prior distribution (i.e., class prior) of y based on X, $P(y|X_i)$ is the partial posterior distribution (i.e., partial class posterior) of y given X_i, $P(X_{i+1})$ is the prior distribution (i.e., observation prior) of X_{i+1} based on $\bigcap_{\mu=0}^{i} X_\mu$, $P(X_i| \bigcap_{\mu=0}^{i} X_\mu)$ is the posterior distribution

(i.e., observation posterior) of X_i given $\bigcap_{\mu=0}^{i} X_\mu$, and n is the number of features vectors.

The causal and mutual action can be distinguished in the formula as follows;

$$C_i \triangleq P(y|X_i) \quad \text{(causal action)} \tag{6.3}$$

$$M_i \triangleq \frac{P(X_i|X_\mu)}{P(X_i)} \quad \text{(mutual action)} \tag{6.4}$$

where y is the target (or output) property, X_i is the vector of input properties of action i, C_i is the causal action of agent i on the target, M_i is the mutual action of agent i with respect to neigbhoring agents, and $P(.)$ denotes a probability function.

Applying such representation to proposition 6.2.1 will lead to

$$C_{n+1} = \frac{1}{W_n} \prod_{i=0}^{n} C_i M_i, \quad W_n = P(y)^{n-1} \tag{6.5}$$

Considering Y to be a dynamic variable of X that changes over time, and X is the actual memory space for each value, the tracking objective is to estimate the next value of Y while keeping track of the previous values. The temporal domain is tracked using temporal actions that act on the temporal (or sequential) inputs, so i represents a temporal property in the CoPreMo model. The figure below illustrates the flow of temporal inputs, actions, and output for the tracking model.

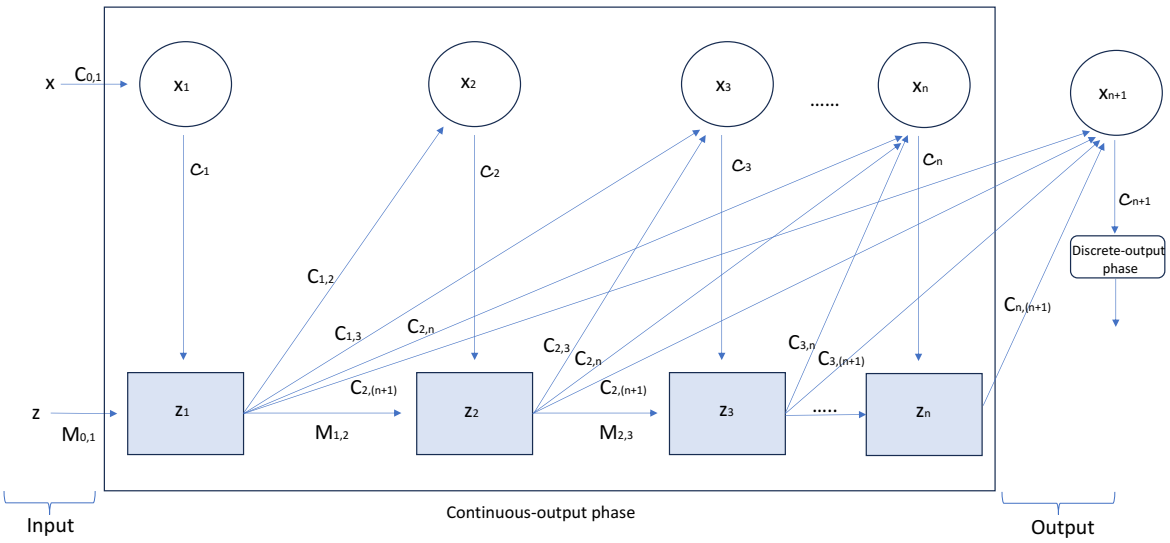


Figure 6.3: Tracking of radar properties using CoPreMo on a 1D input data

In this model, once the radar properties of the target such as range properties are

estimated, they enter the tracking system represented above. In this system, the radar-estimated properties are tracked sequentially as they are measured by the radar. The tracking process consists of predicting the next value of the target in a time series.

To achieve this. a sequence of mutually chained instances of the tracking action is introduced. The training process involves the optimization of both the mutual and causal branches of the model using any collaborative learning approach. After training, the model can be tested with a new sequence of input values. The algorithm for the system implementation is presented in Algorithm 1.

Algorithm 1 Action Hopping Algorithm for target tracking

Require: z (measurement), k (number of event sequence)

Ensure: $C_{n+1} = P(x_{n+1}|z_{1:n})$

$C \sim \mathcal{N}(\mu, \sigma^2)$, $M_1 \leftarrow 1, C_1 \leftarrow 1$, {initialization}

for $n = 1$ to k **do** {action hopping starts}

$C \leftarrow 1, M \leftarrow 1$

for $i = 1$ to n **do** {local estimations}

$C \leftarrow C_i C$ {estimating partial causality}

$M \leftarrow M_i M$ {estimating mutuality}

end for

$C_n \leftarrow C, M_n \leftarrow M$ {local estimate aggregation}

$W_n \leftarrow C_n^{(n-1)}$ {estimating priors}

$C_{n+1} \leftarrow \frac{1}{W_n} C_n M_n$ {global causal update}

end for =0

This variant of CoPreMo where the action hops over the event using sequential instances is considered in this study as an "action hoping" technique. In this car tracking model, the action uses a one-step hoping process, however, more than one-step hopping can also be used, where the action hops over multiple input sequences before making a decision. The important issue with the action hoping algorithm for CoPreMo model is that the action seeks not only to predict the future input state but also to capture the mutuality between the past and present input states.

This model adds to the library of applications that the collaborative framework can be deployed, especially on time series data. The use of this model in real-time applications can save time and complexity due to its simplicity and explainability.

Different performance measures can be used to evaluate a tracking process. This includes statistical measures such as accuracy and error measures used mostly to evaluate the statistical performance of models. However, apart from such statistical performance measures, it is also important to evaluate the system performance of the model, that is the

algorithmic or implementation performance of the model, and common measures under this category include the time and space complexity of the algorithm.

The statistical performance measures used in this study are the accuracy and root mean square error (RMSE)of the tracker. The system performance measure used to evaluate the CoPreMo is the time complexity based on the Big O notation. These performance measures are presented below.

i) Statistical performance measures

The accuracy and RMSE are used in this study as the statistical performance measures. The accuracy measure captures the performance of the tracker at the discrete (classification) output, while the RMS captures the performance of the tracker at the continuous output. The classification is multiclass, with four range classes; ultra-short, short, medium, and long ranges. These are defined as

$$\text{Accuracy} = \frac{P_c}{P_a} \tag{6.6}$$

where P_c is the correct predictions and P_a is all predictions.

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \hat{x}_i)^2}{n}} \tag{6.7}$$

where x_i is the true value and $\hat{x}_i$ is the predicted value.

ii) System performance measures

The time complexity of the CoPreMo algorithm presented in Algorithm 1 can easily be evaluated using the Big O notation. Using this notation, it is easy to estimate the worst-case computational complexity to be $O(n^2)$.

6.3 Experimental Setup

The experiment aims to demonstrate the performance of the proposed CoPreMo model and compare the results with those of conventional models. To validate the performance of our model, we carried out simulation experiments using a road scenario dataset generated from Matlab using the Radar data generator and compared the results with other models.

The experiment consists of an egocar equipped with a radar sensor that moves along a highway and another car that enters the highway from a road inlet until it collides with the egocar as shown in Figure 6.4(a),(b),(d). The radar sensor is placed at the right corner of the egocar as shown in Figure 6.4(c). The simulation was recorded when the egocar

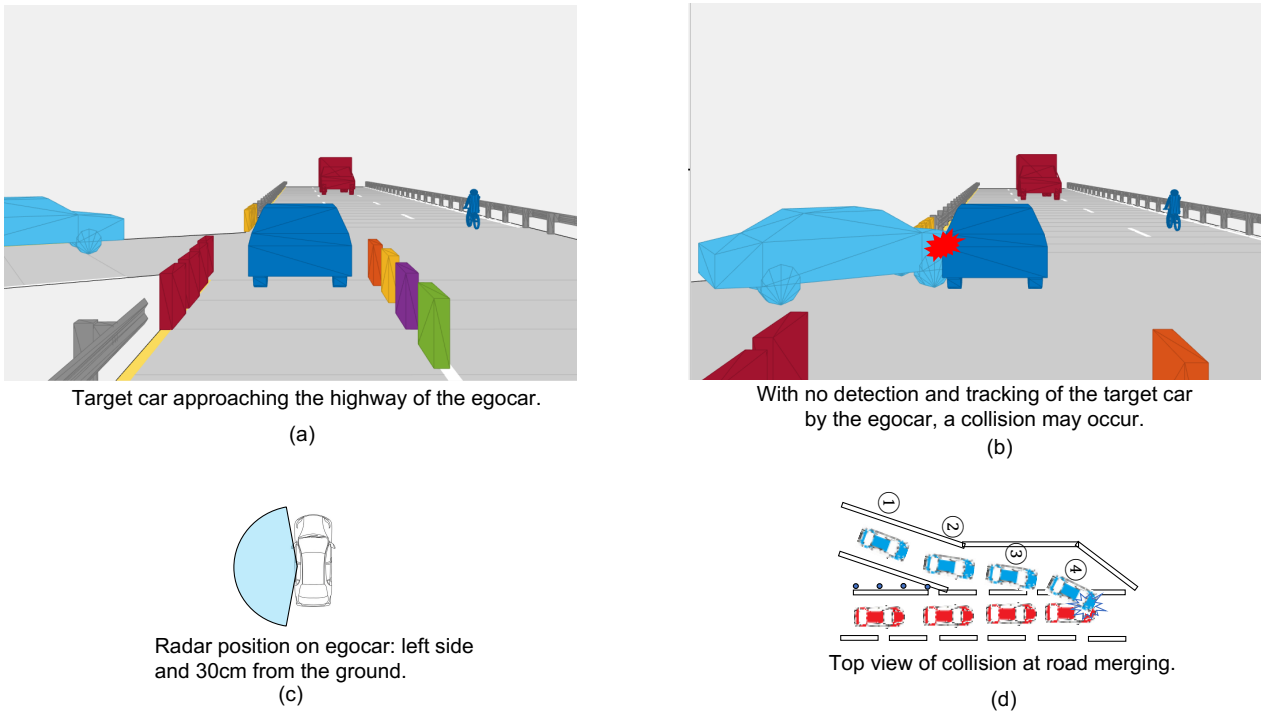


Figure 6.4: Simulation of an ADAS/AD vehicle collision scenario with corner radar detection and tracking.

started observing the target car and the range of the target car as it approached the egocar was recorded.

The dataset for the range estimations was generated using the Matlab Radar data generator. The total number of range measurement points of the target estimated by the radar when the target moves under the radar's field of view is 541 points. These points represent the observation values of the radar. The goal of the tracker is to predict the next measurement value given the current measurement value of this dataset. This was sent to the tracking different tracking models and the results are presented in the next section.

6.4 Results and Discussions

After running the simulation, the performance results for both the continuous-output and discrete-output phases of the proposed and baseline predictive time series tracking models are presented in Table 6.1.

Table 6.1: Performance and numerical comparison.

Models	**Accuracy (%)**	**RMSE**	**Complexity**
BF	91.37	0.51	meduim
KF	93.42	0.48	low
EKF	97.95	0.31	meduim
GSF	98.31	0.26	high
CoPreMo	**98.81**	**0.21**	meduim

From the results, the BF using direct application of Gaussian distribution has an accuracy far less than the KF, EKF, GSF, and the two CoPreMo models. The CoPreMo model performs better than both BF and KF in all the measures. One of the reasons for such high performance of the CoPreMo model is the use of multiple causal values to determine the global output and the mutual value consideration between these values.

..

Chapter 7
Conclusion

This study presented a learning model using a collaborative machine-learning approach. The agents established a mutual value exchange mechanism among the agents on their separate causal actions on the target.

The drawback of the proposed model is that it requires higher design complexity than conventional feedforward neural networks, making it more computationally expensive. Nevertheless, apart from having a better generalization on a given dataset, joining two networks to collaborate syn- synchronously or asynchronously on a target is an important feature of our model. This is because such a collaborative technique can be used to solve multi-label and multi-tasking problems in artificial intelligence by attributing each agent in the collaborative network to a single target (or task) and letting them share mutual values with other agents during learning to enhance their performance on the target.

Finally, the proposed collaborative machine learning model, where multiple learning agents are used to learn and predict a target in a heterogeneous environment, may be deployed in different applications where heterogeneity is present such as in signal processing, natural language processing, and autonomous driving.

”Many hands do light work, but if they are mutually related, they do better work”.

Bibliography

[1] Nancy Buchan, Gianluca Grimalda, Rick Wilson, Marilynn Brewer, Enrique Fatas, and M. Foddy. Globalization and human cooperation. *Proceedings of the National Academy of Sciences of the United States of America*, 106:4138–42, 04 2009.

[2] Francesco Malvestuto and C. Zuffada. The classification problem with semantically heterogeneous data. pages 157–176, 06 1988.

[3] C. Robert Taylor. Dynamic programming and the curses of dimensionality. 1994.

[4] Christopher Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. 10 2007.

[5] Lidong Wang. Heterogeneous data and big data analytics. *Automatic Control and Information Sciences*, 3:8–15, 08 2017.

[6] Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. Weakly-supervised neural text classification. Association for Computing Machinery, 2018.

[7] Wenting Li, Shangbing Gao, Hong Zhou, Zihe Huang, Kewen Zhang, and Wei Li. The automatic text classification method based on bert and feature union. pages 774–777, 12 2019.

[8] Mona Alduailij, Qazi Waqas Khan, Muhammad Tahir, Muhammad Sardaraz, Mai Alduailij, and Fazila Malik. Machine-learning-based ddos attack detection using mutual information and random forest feature importance method. *Symmetry*, 14(6), 2022.

[9] Zhenpeng Liu, Yihang Wang, Fan Feng, Yifan Liu, Zelin Li, and Yawei Shan. A ddos detection method based on feature engineering and machine learning in software-defined networks. *Sensors*, 23(13), 2023.

[10] Euclides Neto, Sajjad Dadkhah, and Ali Ghorbani. Collaborative ddos detection in distributed multi-tenant iot using federated learning. pages 1–10, 08 2022.

[11] Changshun Du and Lei Huang. Text classification research with attention-based recurrent neural networks. *International Journal of Computers Communications Control*, 13:50, 02 2018.

[12] Mazhar Awan, Umar Farooq, Hafiz Babar, Awais Yasin, Haitham Nobanee, Muzammil Hussain, Owais Hakeem, and Azlan Zain. Real-time ddos attack detection system using big data approach. *Sustainability*, 13:10743, 09 2021.

[13] Muhammad Akhter, Zheng Jiangbin, Syed Irfan Naqvi, Mohammed Abdelmajeed, Atif Mehmood, and Muhammad Tariq Sadiq. Document-level text classification using single-layer multisize filters convolutional neural network. *IEEE Access*, PP:1–1, 02 2020.

[14] Zie Eya Ekolle, Ryuji Kohno, and Hideki Ochiai. A mathematical theory of knowledge for intelligent agents. *Preprint*, 2023.

[15] Vijay Vemuri. The hundred-page machine learning book. *Journal of Information Technology Case and Application Research*, 22, 05 2020.

[16] Pratap Sen, Mahimarnab Hajra, and Mitadru Ghosh. *Supervised Classification Algorithms in Machine Learning: A Survey and Review*, pages 99–111. 01 2020.

[17] C. Drew and M. W. John. *Machine Learning for Hackers: Case Studies and Algorithms to Get You Started.* 2012.

[18] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*, volume 82. 01 2003.

[19] G. Hughes. On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory*, 14(1):55–63, 1968.

[20] Geoffrey Hinton. Deep belief networks. *Scholarpedia*, 4:5947, 01 2009.

[21] Asja Fischer and Christian Igel. Training restricted boltzmann machines: An introduction. *Pattern Recognition*, 47:25–39, 01 2014.

[22] Geoffrey Hinton. Article training products of experts by minimizing contrastive divergence. *Neural computation*, 14:1771–800, 09 2002.

[23] V. Chamola, V. Hassija, V. Gupta, and M. Guizani. A comprehensive review of the COVID-19 pandemic and the role of IoT, drones, AI, blockchain, and 5G in managing its impact. 8:90225–90265, 2020.

[24] M. Elhoseny, N. Thilakarathne, M. Alghamdi, R. Mahendran, A. Gardezi, H. Weerasinghe, and A. Welhenge. Security and privacy issues in medical Internet of things: Overview, countermeasures, challenges and future directions. *Sustainability*, 13:11645, October 2021.

[25] Jin Wang, Yaqiong Yang, Tian Wang, Robert Simon Sherratt, and Jingyu Zhang. Big data service architecture: A survey. *Journal of Internet Technology*, 21:393–405, 2020.

[26] R. Amine, S. Sendra, J. Lloret, and A. Oumnad. Internet of things for measuring human activities in ambient assisted living and e-Health. *Netw. Protocols and Algorithms*, 8:15, December 2016.

[27] P. Malhotra, Y. Singh, P. Anand, P. Bangotra, D.and Singh, and W. Hong. Internet of things: Evolution, concerns and security challenges. *Sensors*, 21:1809, March 2021.

[28] A. Djenna and D. E. Saidouni. Cyber attacks classification in iot-based-healthcare infrastructure. In *Proc. 2nd Cyber Secur. Netw. Conf. (CSNet)*, pages 1–4, Paris, France, October 2018.

[29] Mohamed Elhoseny, Navod Thilakarathne, Mohammed Alghamdi, Rakesh Mahendran, Akber Gardezi, Hesiri Weerasinghe, and Anuradhi Welhenge. Security and privacy issues in medical internet of things: Overview, countermeasures, challenges and future directions. *Sustainability*, 13:11645, 10 2021.

[30] T. Martin. On the need for collaborative intelligence in cybersecurity. *Electronics*, 11:2067, June 2022.

[31] M. Ahmed, S. Byreddy, A. Nutakki, L. Sikos, and P. Haskell-Dowland. ECU-IoHT: A dataset for analyzing cyberattacks in internet of health things. *Ad Hoc Networks*, 122:102621, November 2021.

[32] Cristiano André da Costa, Cristian Pasluosta, Bjoern Eskofier, Denise Bandeira, and Rodrigo Righi. Internet of health things: Toward intelligent vital signs monitoring in hospital wards. *Artificial Intelligence in Medicine*, 89, 06 2018.

[33] Faisal Hussain, Syed Ghazanfar Abbas, Ghalib A. Shah, Ivan Miguel Pires, Ubaid U. Fayyaz, Farrukh Shahzad, Nuno M. Garcia, and Eftim Zdravevski. A framework for malicious traffic detection in iot healthcare environment. *Sensors*, 21(9), 2021.

[34] Syed Abbas, Faisal Hussain, Atiq Rehman, Usama Ubaid, Farrukh Shahzad, and Ghalib Shah. Iot-flock: An open-source framework for iot traffic generation. *2020 International Conference on Emerging Trends in Smart Technologies (ICETST)*, pages 1–6, 03 2020.

[35] Abdullah Alsaedi, Nour Moustafa, Zahir Tari, Abdun Mahmood, and Adnan Anwar. Ton_iot telemetry dataset : A new generation dataset of iot and iiot for data-driven intrusion detection systems. *IEEE Access*, 8, 09 2020.

[36] Ilhan Firat Kilincer, Fatih Ertam, Abdulkadir Sengur, Ru-San Tan, and U. Rajendra Acharya. Automated detection of cybersecurity attacks in healthcare systems with recursive feature elimination and multilayer perceptron optimization. *Biocybernetics and Biomedical Engineering*, 43:30–41, 2023.

[37] Izhar Ahmed Khan, Nour Moustafa, Imran Razzak, M. Tanveer, Dechang Pi, Yue Pan, and Bakht Sher Ali. Xsru-iomt: Explainable simple recurrent units for threat detection in internet of medical things networks. *Future Generation Computer Systems*, 127:181–193, 2022.

[38] Georgios Zachos, Ismael Essop, Georgios Mantas, Kyriakos Porfyrakis, Jos© C. Ribeiro, and Jonathan Rodriguez. An anomaly-based intrusion detection system for internet of medical things networks. *Electronics*, 10(21), 2021.

[39] Varun Dogra, Sahil Verma, Kavita ., Pushpita Chatterjee, Jana Shafi, Choi Jaeyoung, and Muhammad Fazal Ijaz. A complete process of text classification system using state-of-the-art nlp models. *Computational Intelligence and Neuroscience*, 2022:1–26, 06 2022.

[40] E Figure. Twitter us airline sentiment dataset. *Available online: https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment*, 2018.

[41] R Harun. Research papers dataset. *Available online: https://www.kaggle.com/datasets/harunshimanto/research-paper*, 2018.

[42] T.A. Almeida, J.M.G. Hidalgo, and A. Yamakami. Contributions to the study of sms spam filtering: New collection and results. *In Proceedings of the 11th ACM Symposium on Document Engineering, Mountain View, CA, USA, 19–23 September 2011*, 09 2021.

[43] B Keshav. Autonomous cars: Past, present and future - a review of the developments in the last century, the present scenario and the expected future of autonomous vehicle technology. *ICINCO 2015 - 12th International Conference on Informatics in Control, Automation and Robotics*, pages 191–198, 2015.

Appendix A
Publications

Journals

The research activity related to this thesis has so far produced the following journal publications (ordered by publication date).

(1) Zie Eya Ekolle, Ryuji Kohno, "GenCo: A Generative Learning Model for Heterogeneous Text Classification Based on Collaborative Partial Classifications", Applied Sciences, MDPI, 2023.

(2) Zie Eya Ekolle, Hideki Ochiai, Ryuji Kohno, "Collabo: A Collaborative Machine Learning Model and its Application to the Security of Heterogeneous Medical Data in an IoT Network", IEEE Access, 2023.

In more detail (and with regard to the contributions of this thesis), the following manuscript is under peer-review for publication.

- Zie Eya Ekolle, Ryuji Kohno, "CoPreMo: A Collaborative Predictive Model in Time Series and its Application to Target Tracking in ADAS/AD Radar", IEEE Transactions on Intelligent Vehicles (under review).

Conferences

The research activity related to this thesis has so far produced the following conference publications (ordered by publication date).

(1) Zie Eya Ekolle, Ryuji Kohno, Hideki Ochiai, "A Distributed Cybersecurity Solution in an IoMT Network Using a Multi-target Federated Learning", 2023 IEEE 17th International Symposium on Medical Information and Communication Technology (ISMICT), Lincoln, NE, USA, May 10-12, 2023.

(2) Zie Eya Ekolle, Ryuji Kohno, Hideki Ochiai, Sawada Sadamasa, Ishikawa Ikenji, Ikeda Hiroshi, Ohashi Naomi, "A Reliable 79GHz Band Ultra-Short Range Radar for ADAS/AD Vehicles Using FMCW Technology", 2023 IEEE International Automated Vehicle Validation Conference, Austin, Texas, USA, October 16-18, 2023.

(3) Zie Eya Ekolle, Ryuji Kohno, Mohammad Ghavami, "A Reliable Antenna Array for the 28GHz mmWave Band in a 5G Massive MIMO Communication", 26th International Symposium on Wireless Personal Multimedia Communications (WPMC 2023), Tempa, Florida, USA, November 19 - 22, 2023.