INFORMATION FUSION USING CONVOLUTIONAL TRANSFORM LEARNING

By

POOJA GUPTA

(PhD18018)

Under the supervision of Prof. Angshul Majumdar

COMPUTER SCIENCE AND ENGINEERING

INDRAPRASTHA INSTITUTE OF INFORMATION TECHNOLOGY DELHI

NEW DELHI– 110020

SEPTEMBER, 2023

Information Fusion using Convolutional Transform Learning

By

Pooja Gupta

PhD18018

A Thesis

submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Computer Science and Engineering

Indraprastha Institute of Information Technology Delhi

New Delhi– 110020

September, 2023

Certificate

This is to certify that the thesis titled *Information Fusion using Convolutional Transform Learning* being submitted by *Pooja Gupta* to the Indraprastha Institute of Information Technology Delhi, for the award of the degree of Doctor of Philosophy, is an original research work carried out by her under my supervision. In my opinion, the thesis has reached the standard fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree or diploma.

September, 2023

Prof. Angshul Majumdar

Indraprastha Institute of Information Technology Delhi

New Delhi 110020

Abstract

There are many real-world problems pertaining to the need for the fusion of information from multiple sources. Consider, for example, the problem of demand forecasting that requires estimating the power consumption at a future point given the available information till the current instant. At the building level forecasting, the inputs are usually power consumption, weather(temperature, humidity), and occupancy. This is a crucial problem in smart grids that ranges from planning electricity generation to preventing non-technical losses. Likewise, many such real-world examples can be cast as multi-channel information fusion based problems. Thus, we need the techniques whereby this varied nature of information from multiple sources can be combined/fused to predict some value(s) that can contribute significantly to future decision making.

A bountiful of techniques have been proposed so far for multi-channel fusion, yet hardly any of them have been addressed as an end-to-end fusion formulation. Few of such solutions are based on techniques that include - Deep learning and Statistical Machine Learning (SML) algorithms. However, existing solutions related to deep learning paradigms involve Convolutional Neural Network (CNN). The latter might not guarantee distinct filters and hence, quality representations might not be obtained that could lead to redundancy. Secondly, CNNs are supervised and, therefore, require large labelled datasets that are not readily available in every other domain. Lastly, SML algorithms are largely prone to overfitting as these heavily rely on quality of features input. Thus, end-to-end, multi-channel, both unsupervised and supervised Convolutional Transform Learning (CTL) based solutions are proposed that bridges all the gaps. The problems targeted lie under multiple domains including financial, biomedical and multiview image and text datasets.

Firstly, this dissertation proposes unsupervised multi-channel fusion solutions to the problems in the financial domain - stock trading(trend prediction/classification) and stock forecasting(price prediction/regression) both of which include

time-series data. It preserves the true nature of time-series as univariate instead of frameworks treating them as 2D matrix/image. Also, the given solution is highly efficient in terms of training a single framework single framework and obtaining features that can be utilized for both classification and regression tasks. The latter benefit cannot be achieved with CNNs.

Secondly, multiple information fusion problems are solved by giving supervised frameworks based on CTL and deep learning paradigms. Specifically, one of the frameworks is proposed to cater to the problem of stock trading that eliminates the issue of dead ReLU and guarantees representations that are more diverse helping in obtaining better performance over the state-of-art techniques. The latter has been validated via fair comparison with CNN where the proposed method supersedes it. Next, an information fusion solution is given that is supervised jointly trained and optimized approach based on CTL and Decision Forest (DF) for predicting Drug-Drug Interactions that could lead to Adverse Drug Reactions (ADRs) instead of utilizing them in a piecemeal fashion.

Lastly, this thesis contributes to solve multiview clustering fusion problem handling the challenge of data-constrained scenarios. It involves the multiview datasets under image and text categories. A joint optimization of Deep CTL (DCTL) and K-Means clustering is proposed. It avoids the piecemeal approach and learns representations from clustering perspective with the help of K-Means clustering loss.

Dedication

I dedicate my work to my family and friends. I am grateful to my loving parents, grandparents, brother and in-laws whose words of encouragement and push for tenacity ring in my ears - "Where there's will, there's a way !". Special thanks to my husband Ankur for his whole-hearted belief in me, constantly motivating me, and for never leaving my side in this journey.

Acknowledgements

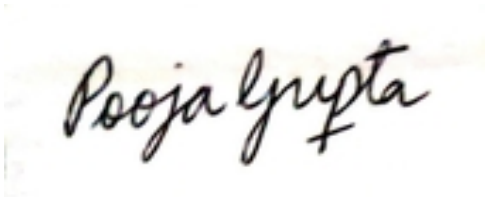
I owe a big thanks to my advisor Prof. Angshul Majumdar who is behind this day when I can call myself a researcher and can prepend Dr. with my name. His constant guidance, support and friendly behaviour have helped to sail through in my journey of research. He has always given an outstanding environment of research and infrastructure, his time and efforts for brainstorming discussions. Prof. Angshul has always lend his ears to my problems in tough times and helped me to overcome them with his words of encouragement and motivation. I am truly blessed to have him as a supervisor, a mentor and a friend.

I would like to thank Indraprastha Institute of Information Technology Delhi that gave me an opportunity to be part of it as a researcher. Also, the institute has given excellent infrastructure and proactively solved any of the issues in access to any necessary things , especially, Mr. Adarsh Kumar Agarwal sir from IT helpdesk and others too. I also want to thank my Internal Committee members Dr. Pushpendra Singh and Dr. Sanat Biswas and collaborators Dr. Emilie Chouzenoux, Dr. Giovanni Chierchia and Dr. Ronita Bardhan for providing their insightful comments in the research assessments and collaboration works respectively.

Next, I want to thank my Grandparents, Late Shri. Ram Nath Gupta and Late Smt. Shanti Devi Gupta for continuously showering blessings. I am thankful to my parents (Dr. Satish Chandra Gupta and Mrs. Sunita Gupta) for working hard to provide me with the privilege of having a good life and attaining this prestigious degree. I am also grateful to my in-laws (Mr. Ashok Kumar Gupta and Mrs. Manju Gupta) for understanding my ambitions and supporting me post-marriage during my journey. I especially want to thank my husband, Ankur, for his care, patience, encouragement, and unwavering support throughout this journey. My daughter Vedanshi, born recently, has been an integral part of this journey, for she has been my lucky charm. She has always made me feel alive in the time of melancholy, for which I feel really blessed. I am also grateful to my

brother Rishabh and brother-in-law Akshay Kumar Gupta for their never-ending love, motivation and support.

I also thank my labmates - Jyoti Maggu, Aanchal Mongia, Priyadarshini Rai, Shalini Sharma, Anurag Goel, Shikha Singh, Megha Gupta Gaur, Vanika Singhal and Kriti Gupta for their companionship and never-ending tea-time stories. I humbly acknowledge their help and support. I also present my sincere gratitude to my friends from other labs Anand Singh, Gunjan Singh, Dhananjay Kimothi, Charul Paliwal, Saurabh Aggarwal, Neetesh Pandey, Ashwini Teertha, Smriti Chawla and Sarita for always supporting me. Last but not the least I thank my all time friends Naina Gupta, Sonal Goel, Shalini Sheoran, Pragati Sharma, Pradyumn Nand, Mona Nandwani, Love Chopra, Prachi Luthra Chopra, Rachit Rakhyani and Bani Rakhyani for standing by my side always.



(POOJA GUPTA)

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Information Fusion (IF) is an advanced process that involves estimation and empowers users to asses complex situations more efficiently, effectively and accurately. It combines information from multiple sources that can be massive, diverse and sometimes conflicting as well. This integration produces specific and comprehensive unified estimates about an entity, activity or event. According to [2], IF is defined as “the study of efficient methods for automatically or semi-automatically transforming information from different sources and points in time into a representation that provides effective support for human or automated decision making" . Many real-world domains raise problems pertaining to the need for the fusion of information from multiple sources.

Let us consider the demand forecasting problem that requires estimating the power consumption at a future point by accounting for the available information until the current instant. Usually, in this respect, the inputs at the building level forecasting are power consumption, weather (temperature, humidity), and

occupancy etc. It is pertinent to solve this problem as it is a crucial aspect in smart grids that ranges from planning electricity generation to preventing non-technical losses. Next, we consider biomedical signal analysis where IF is required. For example, the problem of blood pressure estimation. The inputs are usually from two sources, namely the electrocardiogram (ECG) and pulsepleithismogram (PPG) [3], and the goal is to estimate the systolic and diastolic pressures.

Transportation is also a domain that needs the fusion of information from many sources to build intelligent transportation systems (ITS) [4, 5]. It is essential to improve safety of a passenger, reduce transportation time and fuel consumption, etc. In the same domain, the work [6] deals with the problem of forecasting the taxi demand in the event areas. It is done by fusing the publicly available data and time-series data using deep learning techniques.

Image fusion is another area where the information from two or more images of an object has to be integrated into a single image that is more informative and appropriate for visual perception or computer analysis. It is significantly applied in medical imaging. For example, to improve the functional and spatial information content of the PET images, the fusion of Magnetic Resonance Imaging (MRI) and Positron Emission Tomography (PET) images using Intensity Hue Saturation (IHS) and Retina-Inspired Models (RIM) fusion methods is performed [7]. Multi-sensor video is also a domain that requires multi-channel data fusion applied in the medical domain. It uses the fused video displays and scanpath assessment for the visible and infrared side-by-side[8].

Similarly, there are other domains that we will not discuss at length but briefly mention where IF plays its role. Opinion mining based on sentiment analysis [9], Stock Price prediction [10], drug-drug interaction [11], human activity recognition [12] etc. Thus, IF is a field that offers a plethora of opportunities to solve many impactful real-world problems.

1.1 Problem Statement

This research dissertation aims to propose efficient multi-channel fusion frameworks that learn better representations for solving problems in the analysis domain. There could be supervised and unsupervised learning tasks finding applications in n-Dimensional data domains. The problems targeted under the supervised category are regression and classification and clustering for unsupervised. The aim is to offer quick decision-making to the practitioner while dealing with information from multiple sources.

1.2 Background

IF integrates heterogeneous data from multiple sources to learn representations that can lead to effective decision-making in future events. The challenge comes here, like Data imperfection, inconsistency, confliction, alignment and correlation, heterogeneity, etc. [13]. Thus, IF is an area that needs solutions to overcome these challenges for different applications. A range of methods have been proposed for solving the problems in IF, from probabilistic methods

and statistical machine learning to deep learning. We will briefly discuss these techniques covering some problems in different application domains.

1.2.1 Probabilistic approach

Several studies have used probabilistic methods. As mentioned previously, IF finds excellent application in Intelligent Transportation Systems (ITS). According to one of the fusion-based ITS surveys [14], most of the studies had used probabilistic-based fusion methods since 2011 with a percentage of 46.29% (i.e., 81 out of 135 articles studied in the survey were based on probabilistic based fusion). It is worth mentioning that these include Kalman Filter (KF) algorithm and its variations, e.g., Extended Kalman Filter (EKF) [15–21], Sequential Kalman Filter (SKF) [22] etc. The kinds of applications under ITS covered here concerning KF are car or vehicle positioning [15, 22] in a smart city, vehicle localization [16, 19, 20], moving object detection and tracking [18], navigation [21] etc.

Opinion Mining (OM) is another area of application where IF finds its scope to solve real-world problems. OM is the technique that involves the task of extracting opinions from unstructured text by combining techniques from Natural Language Processing (NLP) and Computer Science. Here, also probabilistic models are applied [23, 24]. The study in [25] presents an “Enterprise IF" framework that exploits many techniques to better understand the impact on an enterprise’s business. The latter includes client feedback and any noteworthy news about events that could affect it. Also, sometimes involve corporate’s

data for analysis. Thus, such a framework depends on multiple sources of information - news sourced from platforms like Twitter and feedback sourced from comments on discussion boards and Really Simple Syndication (RSS) feeds from specific blogs. For this purpose, they use a “blackboard architecture" described in [23]. It is a belief network with nodes representing propositions with associated probability distributions and edges denoting different conditions on nodes. The study’s authors observed a dip in sales of a given product after higher negative feedbacks. They stated that even though their analysis was ex-post, the unstructured data mining synchronized with sales data could have provided insights to perform better marketing campaigns and find a better market niche for the observed product.

Another sector that uses probabilistic Bayesian models is healthcare. With the help of Bayesian frameworks, functional MRI, multi-variate decoding and psychophysics, the authors in [26] demonstrated Bayesian causal inference through hierarchical multi-sensory processes in the human brain. The Internet of Things (IoT) is also an area where IF finds its application. Consider a smart home, i.e., where data is collected through different sensors installed in the home. One such problem that can be solved is knowing about the person’s occupancy while preserving privacy. In the study [27], sensors data, including temperature, humidity, light and CO_2 are used to detect the occupancy in a room. The authors considered each kind of readings’ Probability Density Functions (PDF) and calculated the Probability Mass Assignment (PMA) for each parameter y to be in class x. Next, Dempster’s combination rule was applied to combine the

PMA value for a final decision [27]. However, Dempster-Shafer's theory poses difficulty in estimating mass due to which its applications are limited.

The drawback with the probabilistic models is that it is difficult to obtain a density function and define priori probabilities. Also, when dealing with complex and multi-variate data, we have to settle for limited performance. Further, we cannot handle uncertainty with such solutions [28].

1.2.2 Machine Learning-based Frameworks

Traditional machine learning algorithms have also been extensively used to solve many IF-based problems. In the study [29], the task was to classify the incorrect driving behavior using multiple inputs, including the driver's driving operation behavior, steering wheel angle, brake force, and throttle position. Also, it considers road conditions and then classifies using these inputs via the Adaboost algorithm. In another study [30], activity recognition is performed via fusion at two levels - feature and score fusion levels through Naive Bayes Algorithm. One of the applications of the ITS and IOT is the vacant parking spot detection problem in urban environments. In view of the same, the work in [31] employs the fusion of the information from small-scale sensor-based detectors with that obtained from exploiting the widely-deployed video surveillance camera networks. This framework again utilizes traditional ML algorithms k-Nearest Neighbors (kNN) and Support Vector Machine (SVM) on Histograms of Oriented Gradients (HOG) and Gabor histograms features extracted.

Sentiment classification is more challenging than document topic classification as the latter has specific keywords that do not require context /emotion to be understood as in the former case. In sentiment classification also, works exist where fusion is performed via Naive Bayes, Maximum Entropy Classifier and SVM where SVM superseded the other two [32]. In the healthcare sector, consider analysing the EEG (Electroencephalogram) signal in the case of stroke patients and classifying the stroke as ischemic stroke and hemorrhagic stroke. The method in [33] is a multi-feature fusion method that combines wavelet packet energy, fuzzy entropy and hierarchical theory. Further, SVM, Decision Tree (DT) and Random Decision Forest (RDF) are used as the stroke signal classification models.

Fault detection in motors also requires the fusion of information. In this regard, Banerjee et al. [34] proposed a hybrid method for fault detection based on multi-sensor data fusion with SVM, Short Term Fourier Transform (STFT) and a time duration-based observer model. License Plate (LP) detection technique is another fusion-based problem. It is based on multistage IF adopted for reducing the high false alarm rate in the conventional Adaboost detector [35]. The latter is enhanced via a color-checking module and an SVM detector that checks the image patch for LP. Another domain of the IF application is finance, specifically, the stock market. One such event is to predict stock price movement. In study [36], authors gathered historical stock market data and derived technical indicators followed by integration of Wikipedia hits and Google news data to prepare a rich knowledge base. Using this data, the authors generated features and utilized

three ML models - DT, SVM, and Artificial Neural Networks (ANNs) for stock trend prediction. Also, an intelligent trading-based expert tool was presented to assist the decision-making process on various instruments.

We can see that many domains have used IF based on traditional ML algorithms. However, the issue with traditional ML is overfitting owing to their non-linear mapping and fitting capability. Also, they are highly dependent on the quality of features and hence may not be able to build a relation between input and output variables.

1.2.3 Fuzzy based systems

IF solutions are also based on fuzzy logic. According to one of the surveys [37], under topic identification and trend analysis, specifically, multimodality medical image fusion and fuzzy-based intelligent health and medical systems accounts for 10.06% for Structural Topic Modelling (STM) research. Multi-modal medical image fusion has emerged as a hot topic of research. For the same, the work in [38] proposed a framework based on Structural Patch Decomposition (SPD) and fuzzy logic technology. This framework creates a fused image by adopting a weighted approach as a final step. Next, in Wireless Sensor Networks (WSN), traditional weighted fuzzy logic is not adapted to raw data due to invalid data gathered during data collection in a real-world environment. Thus, to improve upon the same, the work in [39] uses K-Means clustering in addition to fuzzy logic and final fusion is performed using a weighted approach.

Another fuzzy-based fusion approach at the feature level is adopted for intrusion detection in [40] that overcomes the data imperfection challenge of IF. Under ITS, for high-speed heavy vehicles, a Global Positioning System (GPS) based navigation method is developed by authors in [41]. The work used fuzzy logic to fuse the GPS and odometric sensors. Next, under the same category of ITS, to avoid congestion, the fusion framework combines the Inertial Navigation System (INS) and the GPS [42]. It uses Extended KF (EKF) and Input-Delayed Adaptive Neuro-Fuzzy Inference System (IDANFIS) for fusion.

Currently, telemedicine is trending which helps to monitor elderly people in homes and detect if they fall. To detect the same, the study in [43] proposed a data fusion approach based on fuzzy logic with a set of rules directed by medical recommendations. Next, forecasting stock market returns is a challenging task. It is due to the complex nature of the data. The study in [44] developed a framework to predict daily stock price movements. The authors deployed and integrated three data analytical prediction models: ANFIS, Artificial Neural Networks (ANN), and SVMs.

From the above discussion, many fusion approaches in various application domains are based on Fuzzy logic. Nevertheless, the challenge with fuzzy systems is difficulty in setting up rules and membership for certain problems at times.

1.2.4 Deep Learning based fusion approaches

Deep Learning (DL) has been widely used for analyzing multi-channel / multi-sensor signals. It facilitates the automated learning of features versus the hand-crafting or manual selection of features which is required in traditional machine learning algorithms. Thus, it saves the human effort of the latter task mentioned. Also, it can learn the complex mappings between the input and output variables that are otherwise difficult to learn with traditional ML algorithms.

In many DL studies, all the sensors are stacked one after the other to form a matrix using 2-D CNN to analyze the sensor signals. For example, in the study, [12], the authors use the previously mentioned framework with input from multiple body sensors to analyze human activity recognition. It is worth noting that in the study [12], temporal modeling needs to be included. This shortcoming is overcome in [45] where 2-D CNN is used on a time series window. These windows are processed by GRU in the final step and hence time series modeling is incorporated. Nevertheless, there is no explicit fusion framework in all the discussed studies. Though a fusion framework was proposed in [46]; however, the fusion happened at the feature level versus raw signal level like in [12, 45].

Traffic flow prediction is also the use case of information fusion. The study [47] illustrates the use of a deep learning-based encoder–decoder framework with an attention mechanism to capture the correlation between the spatial traffic-flow images' channels. Another study proposes a DL framework using CNNs for object detection from moving vehicle camera images [48]. DL and IF

combination has also been applied in detecting anomaly-based intrusion. The authors presented a framework with three layers neural network as a classifier. They applied five different fusion rules to verify system effectiveness [49].

The fusion has also been observed in solving problems pertaining to the healthcare sector using DL. In [50], the authors designed multi-information fusion convolutional bidirectional Recurrent Neural Networks (RNNs) to detect arrhythmia automatically using ECGs (Electrocardiograms) as input. Additionally, they employed the combination of CNNs and LSTMs for enriching features. They utilized morphological and temporal information from ECG. The authors in another work [51] provided the best adaptation for patients with irregular astigmatism using CNNs. This fusion framework considered multiview Pentacam images in input.

One of the approaches to the problem of video-based action recognition requires IF [52]. It does not take as input audio data for the task, but it proposes a fusion scheme for incorporating temporal information (processed by CNN) and spatial information (also processed by CNN). Experiments were conducted with different levels of early and late fusion. There are studies where multi-channel image dataset fusion has also been investigated. In [53], a fusion scheme is proposed for processing color and depth information (via 3-D and 2-D convolutions, respectively) with the objective of action recognition. In [54], the authors have fused hyperspectral data (high spatial resolution) with Lidar (depth information), subsequently giving better classification results. In [55], an improvement in analysis tasks was observed with the help of the fusion of deeply

learned features (from CNN) with handcrafted features via a fully connected layer.

We see here that several studies include solutions to information fusion-based problems via deep learning technique CNN-based frameworks. However, the issue with CNN is that these are primarily supervised and require large labeled datasets. But, the labeled datasets are only present in abundance for a few domains. Hence, we need unsupervised solutions. Also, the training in CNNs involves learning of filters. However, CNNs may not guarantee distinct filters as any loss function involved generally does not impose any distinctiveness constraint. CNN initializes filters randomly and depends on the non-convergence of backpropagation algorithm to maintain the mutual difference [56]. Thus, there is a possibility that representations/feature maps might be redundant [57]. This has even been shown via experiments discussed later in chapters 2 and 3.

1.3 Datasets Descriptions

This thesis proposes solutions based on CTL for three types of problems under the analysis domain - Supervised, Unsupervised and Clustering. Specifically, the tasks dealt with are - regression, classification and multiview clustering tasks. While proposing solutions, it was the chance to explore and apply them to multiple datasets covering different domains of Information Fusion. It helped us realize that the solutions are generic enough to be applied to the problems pertaining to fusion other than those utilized in this thesis. Thus, the different

datasets used in the problems presented in this thesis are presented below.

1.3.1 National Stock Exchange (NSE) dataset

It is a real dataset from India's National Stock Exchange (NSE). The dataset contains information on 150 symbols between 2014 and 2018; these stocks were chosen after filtering out stocks with less than three years of data. The companies available in the dataset are from various sectors such as IT (e.g., TCS, INFY), automobile (e.g., HEROMOTOCO, TATAMOTORS), bank (e.g., HDFCBANK, ICICIBANK), coal and petroleum (e.g., OIL, ONGC), steel (e.g., JSWSTEEL, TATASTEEL), construction (e.g., ABIRLANUVO, ACC), public sector units (e.g., POWERGRID, GAIL), etc. There are two signals for each sample in the dataset BUY represented as 0 and SELL represented as 1 numerically. The former indicates whether to buy the stock and the latter indicates to sell that stock on any day.

1.3.2 Past 22 years stock data

This dataset consists of 15 Indian stocks that fall under the NSE and the Bombay Stock Exchange (BSE), which are taken from publicly available Yahoo finance symbols data. The stock symbols ending with .NS fall under NSE and with .BO under BSE. The data comprises day-wise readings for the past 22 years, i.e., from 1998 - 2019. It is collected internally using the in-built python module Web and the Yahoo API end-point. At the time of data collection, the year 2019

was still ongoing; hence, the data was only partially available for 2019. Also, there were some missing values for some raw features. Thus, the data for 2019 have not been used in the experiments pertaining to it for simplicity. Further, The dataset includes stocks from multiple sectors, such as Indian consumer products manufacturers (e.g., HINDUNILVR.NS), oil and gas (e.g., CAIRN.NS), pharmaceuticals (e.g., AUROPHARMA.NS, DRREDDY.NS), mining and metal industry (e.g., NATIONALUM.BO). In this dataset, we have three stock signals BUY, HOLD and SELL represented numerically by 0, 1 and 2. The BUY and SELL have the same roles as explained in section 1.3.1 and HOLD signifies that we do nothing on any given day if we are signaled HOLD for that day, i.e., we keep the stock with us and do not buy or sell for that symbol.

1.3.3 Drug-Drug Interaction Data

The DDI data is from Stanford's Biosnap dataset, which contains a network of 1514 DrugBank drugs representing nodes and 48514 drug-drug interactions representing edges. This network of interactions between drugs is approved by the U.S. Food and Drug Administration. It is assumed that all other interactions apart from approved interactions as either known-not-to-interact or unknown. Here, the known-to-interact interactions are numerically represented by 1 and the others by 0. The SMILE values of the drugs are first determined using compound IDs taken from the dataset using DrugBank.ca. Since the SMILE values are not available for all the drugs (retrieved using DrugBank IDs), thus, the number of the drugs in the dataset got reduced to 1368 and, accordingly, the number of

interactions. Further, drugs that have at least 10 known-to-interact interactions with other drugs have been processed. So, there are finally 1059 drugs and their respective interactions considered from the dataset.

Thereafter, the bioactivity descriptors via the Signaturizer tool [58] using the determined smile value of each drug are extracted. This tool provides bioactivity descriptors that encode the physicochemical and structural properties of small molecule drugs covering all the drugs present in Chemical Checker (CC). The latter has further covered the source databases - DrugBank.ca and ChEMBl. It has a pre-trained Siamese Neural Network via which inputting a smile value for the drug, 25 different types of bioactivity descriptors can be inferred for the drugs with little or no information. The descriptors are fixed-length normalized vectors of size 128. There are broadly five categories of bioactivity descriptors labeled as A to E (A: Chemistry, B: Targets, C: Networks, D: Cells, and E: Clinics). Each has five sub-categories marked as A1 to A5, for example, thus 25 different descriptors. Descriptors from A and B broad categories representing a drug's Chemistry and Targets, respectively, are taken. Further, specifically, the A1 and A2 sub-categories from A, representing 2D and 3D fingerprints, and the B1 sub-category from B, representing the mechanism of action of a drug are selected. Since these are three types of bioactivity descriptors out of 25, each having 128 fixed-sized vectors, each drug has $384(128 \times 3)$ features. Thus, the final dataset comprises 1059 unique drugs with 384 bioactivity descriptors/features for each drug and corresponding interactions.

1.3.4 Mutli-view datasets

Here the proposed approach was tested on various multiview clustering datasets listed below:

- 100leaves: It contains one hundred plant species, each with 16 samples per specie. Thus, there are 100 clusters and 1600 total samples. For each sample, shape descriptor, fine scale margin and texture histogram are given [59].
- Amsterdam Library of Object Images (ALOI). ALOI dataset consists of 11025 images of 100 small objects. Every image is represented using four features namely - Color similarity, HSV, RGB, and Haralick features [60].
- Mfeat: Mfeat dataset is from the UCI repository that contains 2000 samples of handwritten digits (0-9). Each image of this dataset is represented using six different features [59].
- WebKB: It consists of 203 web pages with four classes collected from computer science departments of various universities. Each web page is attributed by the page's content, hyperlink's anchor text of the hyperlink and its title text [59].

The complete statistics of all the datasets mentioned above can be referred from Table 1.1.

Table 1.1: Statistics of the considered MVC datasets

Datasets	#Samples	#Classes	#Views
100leaves	1600	100	3
WebKB	203	4	3
Mfeat	2000	10	6
ALOI	11025	100	4

1.4 Research Contributions

This thesis has three main objectives: 1. To propose more accurate algorithms for IF than state of the art; 2. To propose an unsupervised algorithm, unlike CNNs that are largely supervised, thus eliminating the need for large labeled datasets; and 3. To propose methods that ensure that the learned filters are distinct and hence the representations learned are more interpretable after fusion.

Our contributions towards these objectives are as follows:

First, an unsupervised fusion framework has been proposed based on CTL (CTL). The excellent learning ability of convolutional filters for data analysis is well acknowledged [61–66]. The success of convolutive features owes to CNN. However, CNN cannot perform learning tasks in an unsupervised fashion. In recent work, it is shown that such shortcomings can be addressed by adopting a CTL approach, where convolutional filters are learned in an unsupervised manner [56, 67]. Therefore, the proposed framework is (i) a deep version of CTL (DCTL); (ii) an unsupervised fusion formulation taking advantage of the proposed CTL representation; (iii) filters learned are distinct and hence more

interpretable representations are obtained. The proposed techniques, ConFuse [68] and DeConFuse [69], have been applied to the problems of stock forecasting and trading. Comparison with state-of-the-art methods (based on CNN and LSTM network) shows the superiority of our approaches for performing reliable feature extraction.

Second, two supervised frameworks have been proposed that are based on DCTL. The former offer all the benefits of the CTL approach discussed previously. Additionally, the first framework called SuperDeConFuse [57] is such that it facilitated the removal of the non-linear activation located between the multi-channel convolution layers and the fully-connected layers, as well as the one located between the latter and the output layer, thus, handling the problem of a dead neuron. This removal was compensated by introducing a suitable regularization on the aforementioned layer outputs and filters during the training phase. Further, this technique has been applied to the problem of Stock Forecasting. Next, the second supervised framework called - DeConDFFuse [70] is also based on CTL and learns representations guided by joint optimization of Multi-channel DCTL based networks and Decision Forest (DF) rather than a piecemeal approach.

Lastly, a multiview clustering fusion framework based on CTL has been proposed that takes multiview data as input namely DeConFCluster [71]. The framework jointly trains DCTL networks and the K-Means clustering module; thus, the representations are distinct and more effective as these are also guided by K-Means loss. It gives superior clustering performance than the state-of-the-

arts.

For the quick reference, the summary of all the proposed models are given in Table 1.2 and each one's Advantages and Disadvantages are discussed in Table 1.3. More details of each of these models are discussed in subsequent chapters.

Table 1.2: Summary of all proposed frameworks

Learning Technique	**Proposed Model**	**Research objective**	**Dataset Used**	**Application Problem**
Unsupervised	**ConFuse & DeConFuse**	1.,2.,3.	Yahoo Symbol Finance Data	1. Stock Trading 2. Stock Forecasting
Supervised	**SuperDeConFuse**	1.,3.	NSE and BSE Stocks (15)	Stock Trading
	DeConDFFuse	**1.,3.**	Drug Drug Interaction	Drug Drug Interaction Prediction
Unsupervised	**DeConFCluster**	**1.,2.,3.**	1. 100leaves 2. ALOI 3. Mfeat 4. WebKB	Multi View Clustering

Table 1.3: Pros and Cons of all proposed frameworks

Proposed Model	**Advantages**	**Disadvantages**
ConFuse	1. Meets Research objectives	shallow architecture

	2. Avoids Retraining a network for different tasks Classification and Regression	
DeConFuse	same as above 3. It is a deep architecture	-
SuperDeConFuse	1. Meets Research objectives 2. performed better than CNN for the given problem in training than CNN	Takes more time
DeConDFFuse	1. Meets Research objectives 2. Jointly optimizes Decision Forest versus piecemeal approach, thus representations are guided by both CTL based fusion and Decision Forest	Currently handles case when two drugs are administered together when in in real scenario more than two drugs can be used
DeConFCluster	1. Meet Research Objectives 2. avoids additional overhead of learning weights of decoder as it is with encoder-decoder framework used in MVC generally 3. avoided overfitting in data-constrained scenarios where #data instance is low and #classes	less performant for easy-to-cluster datasets compared to benchmarks

	are high 4. Performed well for difficult datasets compared to benchmarks	

1.5 Acronyms

Let's introduce here all the acronyms used in the following chapters for the quick reference to them at one place. Those are as follow:

Table 1.4: Acronyms with full forms used in chapters

Acronym	Full Form
ADAM	Adaptive Moment Estimation
ADR	Adverse Drug Reaction
ALOI	Amsterdam Library of Object Images
ANFIS	Adaptive Neuro-Fuzzy Inference System
ANN	Artificial Neural Network
AR	Annualized Returns
ARCH	Autoregressive Conditional Heteroskedasticity
ARI	Adjusted Rand Index
ARMA	Autoregressive Moving Average
AUC	Area Under Curve
BSE	Bombay Stock Exchange

CC	Chemical Checker
CCA	Canonical Correlation Analysis
CE	Cross Entropy
CMM	Convex Mixture Model
CNN	Convolutional Neural Networks
CoMVC	Contrastive Multi-View Clustering
CTL	Convolutional Transform Learning
DCTL	Deep Convolutional Transform Learning
DCDF	DeConDFFuse
DCKM	Deep Convolutional K-Means Clustering
DDI	Drug-Drug Interaction
DEMVC	Deep Embedded Multiview Clustering
DF	Decision Forest
DL	Deep Learning
DNDF	Deep Neural Decision Forest
DT	Decision Tree
ECG	Electrocardiogram
EEG	Electroencephalogram
EKF	Extended Kalman Filter
EM	Expectation Maximization
EMA	Exponential Moving Average
ETF	Exchange Traded Fund
FCN	Fully Convolutional Network

GARCH	Generalized Autoregressive Conditional Heteroskedasticity
GCN	Graph Convolutional Network
GMC	Graph-based Multi-view Clustering
GNN	Graph Neural Network
GPS	Global Positioning System
GRU	Gated Recurrent Unit
HOG	Histograms of Oriented Gradients
HSV	Hue, Saturation, and Value
IDANFIS	Input-Delayed Adaptive Neuro-Fuzzy Inference System
IF	Information Fusion
IOT	Internet of Things
IHS	Intensity Hue Saturation
IT	Information Technology
ITS	Intelligent Transportation Systems
kNN	k-Nearest Neighbors
KF	Kalman Filter
KG	Knowledge Graphs
KGNN	Knowledge Graph Neural Network
LP	License Plate
LSTM	Long Short Term Memory
MAE	Mean Absolute error
MACD	Moving Average Convergence and Divergence
MCGL	Graph Learning for Multiview Clustering

MFNN	Multi-Filters Neural Networks
ML	Machine Learning
MRI	Magnetic Resonance Imaging
MVC	Multiview Clustering
NAV	Net Asset Value
NLP	Natural Language Processing
NMF	Non-Negative Matrix Factorization
NMI	Normalized Mutual Information
NSE	National Stock Exchange
OM	Opinion Mining
PDF	Probability Density Functions
PET	Positron Emission Tomography
PMA	Probability Mass Assignment
RDF	Random Decision Forest
RGB	Red, Green and Blue
RIM	Retina-Inspired Models
RNN	Recurrent Neural Network
RMSE	Root Mean Squared Error
ReLU	Rectified Linear Unit
ROC	Receiver Operating Characteristic
RRA-MVC	Reconsidering Representation Alignment for Multi-view Clustering
RSS	Really Simple Syndication
SDCF	SuperDeConFuse

SELU	Scaled Exponential Linear Unit
SGD	Stochastic Gradient Descent
SIG	Similarity Induced Graph
SiMVC	Simple baseline Multi-View Clustering
SKF	Sequential Kalman Filtering
SPD	Structural Patch Decomposition
SSL	Self Supervised Learning
STFT	Short Term Fourier Transform
STM	Structural Topic Modelling
SVC	Single View Clustering
SVM	Support Vector Machine
TA	Technical Analysis
TL	Transform Learning
UCI	University of California Irvine
WSN	Wireless Sensor Networks

Chapter 2

Unsupervised multi-channel CTL based fusion frameworks -ConFuse(shallow) and DeConFuse(Deep)

Deep Learning (DL) paradigms currently solve several problems. Most of the frameworks in DL are based on CNNs that are largely supervised. For supervised learning, the labeled data are needed in abundance, which is in dearth for some domains. Also, CNNs cannot perform learning tasks in an unsupervised fashion. The other shortcoming of CNNs, as discussed previously in Chapter 1, is that these may not guarantee learning distinct filters; thus, representations/feature maps might be redundant. It is due to random initialization of filters in CNNs and thus, the latter depends on the non-convergence of the backpropagation algorithm to maintain the mutual difference [56]. This redundancy is further checked experimentally and its details can be referred in this Chapter and Chapter 3 later. Additionally, it has been observed that the problems concerning time-

series data in stock forecasting are treated as a 2-D image matrix versus univariate data, which is the true nature of time-series. We will learn about the said issue in more detail in this chapter in subsequent sections. Thus, there is a need for a solution that can resolve these issues.

This chapter introduces unsupervised fusion frameworks based on Convolutional Transform Learning (CTL). The great learning ability of convolutional filters for data analysis is well acknowledged [61–66]. The convolutive features' success is due to the Convolutional Neural Network (CNN). Nevertheless, as mentioned previously, CNN cannot perform learning tasks in an unsupervised fashion. However, the said shortcoming can be addressed by adopting a recently established Convolutional Transform Learning (CTL) approach, where convolutional filters are learned in an unsupervised fashion. The framework discussed in this chapter is (i) a shallow and a deep versions of the CTL approach; (ii) has an unsupervised fusion formulation taking advantage of the representations learned via CTL and fused via TL; (iii) is a mathematically sounded optimization strategy for performing the learning task; and (iv) learns distinct filters that consequently learn more interpretable non-redundant representations, unlike CNNs.

The proposed frameworks are namely - ConFuse (shallow) and DeConFuse (deep) and are applied to the problems of stock forecasting and trading. Comparison with state-of-the-art methods (based on CNN and LSTM network) shows the superiority of the proposed approaches for performing reliable feature extraction. This chapter is organized into sections, with the first section 2.1 discussing the related work and proposed algorithm in section 2.2. The experimental evalua-

tions and results are discussed in sections 2.3 and 2.4 respectively, followed by discussion in 2.5.

2.1 Literature Review

2.1.1 CNN for Time Series Analysis

Let us briefly review and discuss CNN-based methods for time series analysis. For a more detailed review, the interested reader can peruse [72]. In this section, the main focus are on the studies about stock forecasting as it is the use case for experimental validation.

The traditional choice for processing time series with a neural network is to adopt a recurrent neural network (RNN) architecture. Variants of RNN like Long-Short Term Memory (LSTM) [73] and Gated Recurrent Unit (GRU) [74] have been proposed. However, due to the complexity of training such networks via backpropagation through time, these have been progressively replaced with 1D CNN [75]. For example, in [76], a generic time series analysis framework was built based on LSTM, with assessed performance on the UCR time series classification datasets [77]. The later study from the same group [78], based on 1D CNN, showed considerable improvement over the prior model on the same datasets.

Many studies convert 1D time series data into a matrix form to use 2D CNNs [79–81]. Each matrix column corresponds to a subset of the 1D series

within a given time window, and the resulting matrix is processed as an image. The 2D CNN model has been prevalent in stock forecasting. In [81], the said techniques have been used on stock prices for forecasting. A slightly different input is used in [82]: instead of using the standard stock variables (open, close, high, low and NAV), it uses high frequency data for forecasting major points of inflection in the financial market. In another work [83], a similar approach is used for modeling Exchange Traded Fund (ETF). It has been seen that the 2D CNN model performs the same as LSTM or the standard multi-layer perceptron [84, 85]. The apparent lack of performance improvement in the aforementioned studies may be due to an incorrect choice of CNN model since an inherently 1D time series is modeled as an image.

Another learning paradigm known as Self-Supervised Learning (SSL) based models are also emerging currently when no labels for the data are available. In all such techniques, initially, the data is unsupervised which is eventually turned supervised by predicting the pseudo labels and then training happens. There are few works that utilize and propose solutions based on it for stock trading prediction [86–89]. However, such techniques are resource intense and just like CNNs, these SSL based learning paradigms do not have distinctiveness guarantees.

2.1.2 Convolutional Transform Learning

Convolutional Transform Learning (CTL) has been introduced in a previous seminal paper [56]. Since the proposed framework is based on the said recent

work, it is presented in detail to make it self-contained. CTL learns a set of filters $(t_m)_{1\leq m\leq M}$ operated on observed samples $\left(s^{(k)}\right)_{1\leq k\leq K}$ to generate a set of features $(x_m^{(k)})_{1\leq m\leq M, 1\leq k\leq K}$. Formally, the inherent learning model is expressed through convolution operations defined as

$$(\forall m \in \{1,\ldots,M\}\,, \forall k \in \{1,\ldots,K\}) \qquad t_m * s^{(k)} = x_m^{(k)}. \tag{2.1}$$

Following the original study on transform learning [90], a sparsity penalty was imposed on the features for improving representation ability and limiting overfitting issues. Moreover, the non-negativity constraint was imposed on the features in the same line as CNN models. Training then consisted of learning the data's convolutional filters and representation coefficients. This was expressed as the following optimization problem

$$\begin{aligned}\underset{(t_m)_m,(x_m^{(k)})_{m,k}}{\text{minimize}} \quad & \frac{1}{2}\sum_{k=1}^{K}\sum_{m=1}^{M}\left(\|t_m * s^{(k)} - x_m^{(k)}\|_2^2 + \psi(x_m^{(k)})\right) \\ & + \mu\sum_{m=1}^{M}\|t_m\|_2^2 - \lambda \log\det\left([t_1|\ldots|t_M]\right),\end{aligned} \tag{2.2}$$

where ψ is a suitable penalization function. Note that the regularization term "$\mu\,\|\cdot\|_F^2 - \lambda\log\det$" ensured that the learned filters were distinct, which was not guaranteed in CNN. Let us introduce the matrix notation

$$T * S - X = \begin{bmatrix} t_1 * s^{(1)} - x_1^{(1)} & \ldots & t_M * s^{(1)} - x_M^{(1)} \\ \vdots & \ddots & \vdots \\ t_1 * s^{(K)} - x_1^{(K)} & \ldots & t_M * s^{(K)} - x_M^{(K)} \end{bmatrix} \tag{2.3}$$

where $T = \begin{bmatrix} t_1 & \dots & t_M \end{bmatrix}$, $S = \begin{bmatrix} s^{(1)} & \dots & s^{(K)} \end{bmatrix}^\top$, and $X = \begin{bmatrix} x_1^{(k)} & \dots & x_M^{(k)} \end{bmatrix}_{1 \le k \le K}$. The cost function in Problem (2.2) could be compactly rewritten as[1] as the sum of logarithms of its singular values.

$$F(T, X) = \frac{1}{2} \|T * S - X\|_F^2 + \Psi(X) + \mu \|T\|_F^2 - \lambda \log\det(T), \tag{2.4}$$

where Ψ applied the penalty term ψ column-wise on X.

A local minimizer to (2.4) could be reached efficiently using the alternating proximal algorithm [91–93], which alternates between proximal updates on variables T and X. More precisely, set a Hilbert space $(\mathcal{H}, \|\cdot\|)$, and define the proximity operator [85] at $\tilde{x} \in \mathcal{H}$ of a proper lower-semi-continuous convex function $\varphi : \mathcal{H} \to]-\infty, +\infty]$ as

$$\text{prox}_\varphi(\tilde{x}) = \underset{x \in \mathcal{H}}{\arg\min}\, \varphi(x) + \frac{1}{2} \|x - \tilde{x}\|^2. \tag{2.5}$$

Then, the alternating proximal algorithm reads

$$\begin{aligned} &\text{For } n = 0, 1, \dots \\ &\left| \begin{aligned} T^{[n+1]} &= \text{prox}_{\gamma_1 F(\cdot, X^{[n]})}\left(T^{[n]}\right) \\ X^{[n+1]} &= \text{prox}_{\gamma_2 F(T^{[n+1]}, \cdot)}\left(X^{[n]}\right) \end{aligned} \right. \end{aligned} \tag{2.6}$$

with initializations $T^{[0]}$, $X^{[0]}$ and γ_1, γ_2 positive constants. For more details on the derivations and the convergence guarantees, the readers can refer to [56].

[1] Note that T is not necessarily a square matrix. By abuse of notation, the "log-det" of a rectangular matrix was defined

2.1.3 Updates of T

2.2 Proposed Formulations - ConFuse and DeConFuse

2.2.1 ConFuse: Convolutional Transform Learning Fusion Framework For Multi-Channel Data Analysis

The novel approach *ConFuse*[2] for the unsupervised construction of representation features of multi-channel data is presented in this section. A natural strategy was to learn, for each channel $c \in \{1, \ldots, C\}$, a distinct set of convolutional filters $(T^{(c)})_{1 \leq c \leq C}$ and associated features $(X^{(c)})_{1 \leq c \leq C}$, by solving a CTL-based formulation:

$$\underset{T^{(c)}, X^{(c)}}{\text{minimize}} \ \frac{1}{2} \sum_{k=1}^{K} \left(\|S_k^{(c)} T^{(c)} - X_k^{(c)}\|_F^2 + \Psi(X_k^{(c)}) \right) + \mu \|T^{(c)}\|_F^2 - \lambda \log\det(T^{(c)}). \tag{2.7}$$

Then, the learned channel-wise features were stacked as $X_k = [{X_k^{(1)}}^\top | \ldots | {X_k^{(C)}}^\top]^\top$ for each k, and fused by a transform learning procedure acting as a fully-connected layer:

$$\underset{\widetilde{T}, Z}{\text{minimize}} \ \frac{1}{2} \sum_{k=1}^{K} \|\widetilde{T} X_k - Z_k\|_F^2 + \iota_+(Z) + \mu \|\widetilde{T}\|_F^2 - \lambda \log\det(\widetilde{T}), \tag{2.8}$$

[2]P. Gupta, J. Maggu, A. Majumdar, E. Chouzenoux and G. Chierchia, “ConFuse: Convolutional Transform Learning Fusion Framework For Multi-Channel Data Analysis," 2020 28th European Signal Processing Conference (EUSIPCO), Amsterdam, Netherlands, 2021, pp. 1986-1990, doi: 10.23919/Eusipco47968.2020.9287506.

where $\widetilde{T}$ denoted the fusion stage transform (not assumed to be convolutional), Z is the row-wise concatenation of the fusion stage features $(Z_k)_{1\leq k\leq K}$, and ι_+ is the indicator function for positive orthant, equals to zero if all the entries of Z are non-negative, and $+\infty$ otherwise. Such non-negativity constraint allowed us to avoid trivial solutions.

However, the disjoint resolution of Problems (2.7) and (2.8) might lead to unstable solutions that were too sensitive to initialization. Therefore, an alternative strategy was proposed where all the variables are learned in an end-to-end fashion by solving a joint optimization problem. To this aim, it was relied on the key property that the solution $(\widehat{X}^{(c)})_{1\leq c\leq C}$ of the CTL problem assuming fixed filters $(T^{(c)})_{1\leq c\leq C}$ could be reformulated as the simple application of an element-wise activation function, that is, for every $k \in \{1, \ldots, K\}$,

$$\begin{aligned} \widehat{X}_k(T) &= \left[\widehat{X}_k^{(c)}(T)\right]_{1\leq c\leq C} \\ &= \left[\Phi(S_k^{(c)}T^{(c)})\right]_{1\leq c\leq C}, \end{aligned} \tag{2.9}$$

with Φ the proximity operator of Ψ [94]. For example, if Ψ was the indicator function of the positive orthant, then Φ identified with the famous rectified linear unit (ReLU) activation function. Many other examples are provided in [94]. Consequently, it was proposed to plug Equation (2.9) into Problem (2.8), leading

to the final *ConFuse* formulation:

$$\underset{T,\widetilde{T},Z}{\text{minimize}} \ \frac{1}{2}\sum_{k=1}^{K}\|\widetilde{T}\widehat{X}_k(T)-Z_k\|_F^2+\iota_+(Z)+\mu\|\widetilde{T}\|_F^2$$
$$+\mu\|T\|_F^2-\lambda\Big(\log\det(\widetilde{T})+\sum_{c=1}^{C}\log\det(T^{(c)})\Big). \tag{2.10}$$

Although Problem (2.10) was still nonconvex, this new formulation had two notable advantages. First, it was remarked that, as soon as the involved activation function was smooth, all terms of the cost function in (2.10) were differentiable, except the indicator function. Thus, the accelerated stochastic projected gradient descent, Adam, from [95] could be employed. The latter used automatic differentiation and stochastic approximations to deal with large datasets efficiently. Second, any (sub-)differentiable activation function Φ could be plugged into the proposed model (2.9), for instance, Scaled Exponential Linear Unit (SELU) [96], or Leaky ReLU [97]. This flexibility played a key role in the performance, as shown in the experimental section.

An example of the structure of the learned *ConFuse* architecture is shown in Figure 2.1. Note that the proposed approach was completely unsupervised. Specifically, it replaced supervision by explicitly learning the features Z, on which the non-negativity constraint was imposed to avoid trivial solutions. Regarding the representation filters stacked in matrices $(T, \widetilde{T})$, the log-det regularization imposed a full rank on those. Thus, it helped to enforce the diversity and to prevent the degenerate solution $(T = 0, X = 0, \widetilde{T} = 0, Z = 0)$. The Frobenius regularization ensured that the matrices entries remain bounded.

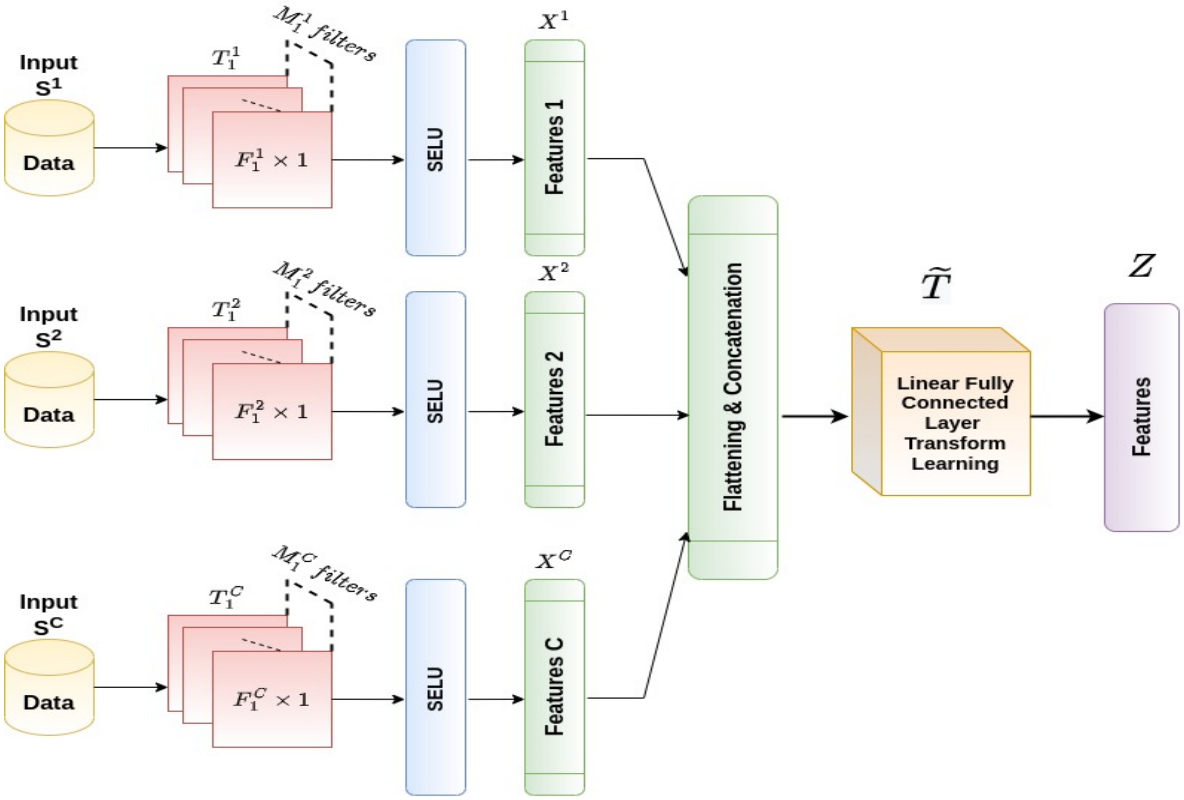


Figure 2.1: General view of the ConFuse architecture. $C = 5$ represents the number of DeepCTL networks/channels, $F_1^c = 5 \times 1$ is the filter size and $M_1^c = 4$ is the number of filters for all the channels.

2.2.2 DeConFuse: a deep convolutional transform-based unsupervised fusion framework

In this framework, the ConFuse architecture was extended with more Convolutional layers based on CTL and called it as - DeConFuse[3]. Here, there were as many Transforms as the number of CTL Layers. Thus, a different set of convolutional filters $T_1^{(c)}, \ldots, T_L^{(c)}$ and features $X_1^{(c)}, \ldots, X_L^{(c)}$ were learned. These learned deep features can be computed by stacking many such layers

$$(\forall \ell \in \{1, \ldots, L-1\}) \qquad X_\ell = \phi_\ell(T_\ell * X_{\ell-1}), \tag{2.11}$$

[3]P. Gupta, J. Maggu, A. Majumdar, E. Chouzenoux and G. Chierchia, “DeConFuse: a deep convolutional transform-based unsupervised fusion framework". EURASIP J. Adv. Signal Process. 2020, 26 (2020). https://doi.org/10.1186/s13634-020-00684-5

where $X_0 = S$ and ϕ_ℓ a given activation function for layer ℓ. Further, these features were processed in the same manner as in the ConFuse architecture, i.e., with fusion transform $\widetilde{T}$ and common representation Z learned subsequently. This led to the joint optimization problem

$$\underset{T,X,\widetilde{T},Z}{\text{minimize}} \underbrace{F_{\text{fusion}}(\widetilde{T}, Z, X) + \sum_{c=1}^{C} F_{\text{conv}}(T_1^{(c)}, \ldots, T_L^{(c)}, X^{(c)} \mid S^{(c)})}_{J(T,X,\widetilde{T},Z)} \tag{2.12}$$

where

$$\begin{aligned} F_{\text{conv}}(T_1, \ldots, T_L, X \mid S) = \frac{1}{2}\|T_L * \phi_{L-1}(T_{L-1} * \ldots \phi_1(T_1 * S)) - X\|_F^2 \\ + \Psi(X) + \sum_{\ell=1}^{L} (\mu||T_\ell||_F^2 - \lambda \log\det(T_\ell)). \end{aligned} \tag{2.13}$$

and

$$\begin{aligned} F_{\text{fusion}}(\widetilde{T}, Z, X) = \frac{1}{2}\Big\| Z - \sum_{c=1}^{C} \text{flat}(X^{(c)})\widetilde{T}_c \Big\|_F^2 + \iota_+(Z) \\ + \sum_{c=1}^{C} \Big(\mu\|\widetilde{T}_c\|_F^2 - \lambda \log\det(\widetilde{T}_c)\Big), \end{aligned} \tag{2.14}$$

where the operator "flat" transformed $X^{(c)}$ into a matrix where each row contained the "flattened" features of a sample. The complete architecture is shown in Figure 2.2.

2.2.3 Optimization Algorithm for the frameworks

As for the solution of Problems (2.10) and (2.12), it was remarked that all terms of the cost function are differentiable, except the indicator function of the non-

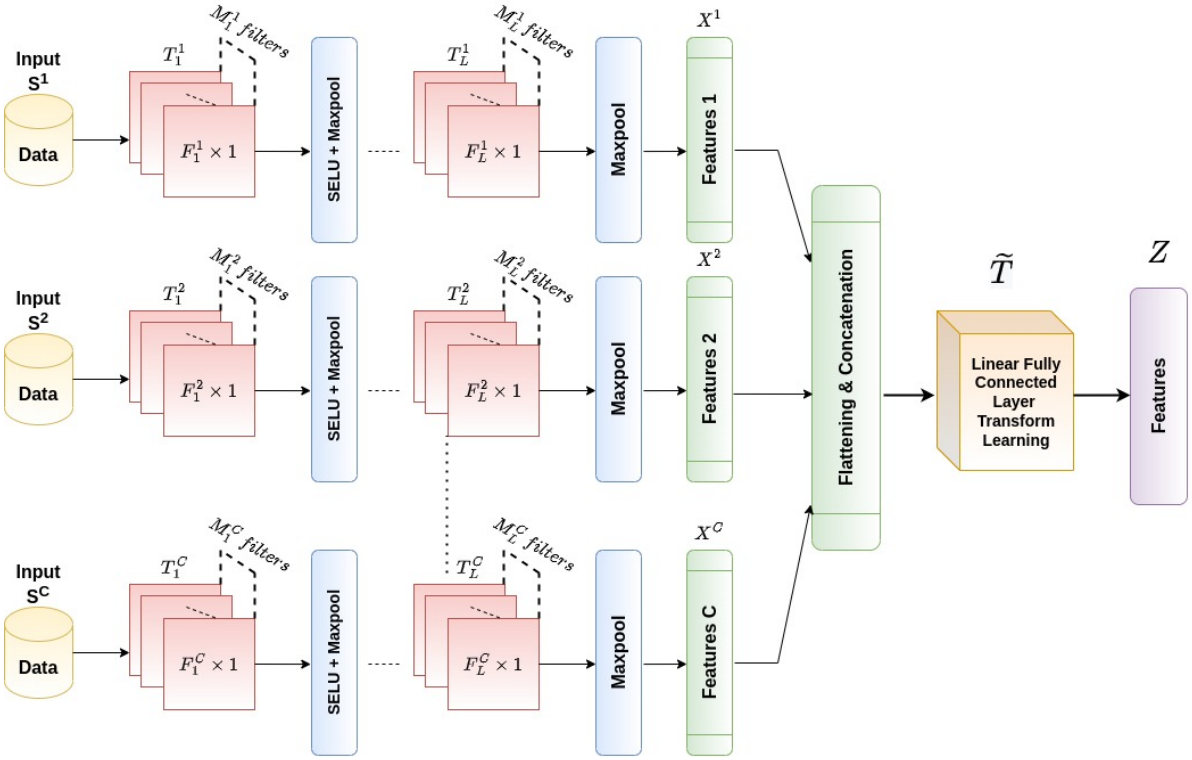


Figure 2.2: General view of the DeConFuse architecture. $C = 5$ represents the number of DeepCTL networks/channels, $L = 2$ is the number of DCTL layers, M_ℓ^c is the filter size and F_ℓ^c is the number of filters of the respective layer ℓ and channel c.

negativity constraint. Therefore, it was possible to find a local minimizer to (2.10) and (2.12) by employing the projected gradient descent, whose iterations read

$$\text{For } n = 0, 1, \ldots$$

$$\left\lfloor \begin{aligned} T^{[n+1]} &= T^{[n]} - \gamma \nabla_T J(T^{[n]}, X^{[n]}, \widetilde{T}^{[n]}, Z^{[n]}) \\ X^{[n+1]} &= \mathcal{P}_+(X^{[n]} - \gamma \nabla_X J(T^{[n]}, X^{[n]}, \widetilde{T}^{[n]}, Z^{[n]})) \\ \widetilde{T}^{[n+1]} &= \widetilde{T}^{[n]} - \gamma \nabla_{\widetilde{T}} J(T^{[n]}, X^{[n]}, \widetilde{T}^{[n]}, Z^{[n]}) \\ Z^{[n+1]} &= \mathcal{P}_+(Z^{[n]} - \gamma \nabla_Z J(T^{[n]}, X^{[n]}, \widetilde{T}^{[n]}, Z^{[n]})) \end{aligned} \right. \tag{2.15}$$

with initialization $T^{[0]}, X^{[0]}, \widetilde{T}^{[0]}, Z^{[0]}$, $\gamma > 0$, and $\mathcal{P}_+ = \max\{\cdot, 0\}$. In practice, the accelerated strategies [98] were used within each step of this algorithm to speed up learning.

There are two notable advantages of the proposed optimization approach. Firstly, it was relied on automatic differentiation [99] and stochastic gradient approximations to efficiently solve Problem (2.10). Secondly, it was not limited to ReLU activation in equations (2.9) and (2.11), but instead, more advanced ones were used, such as SELU [96]. It can be observed from Tables 2.3 and 2.5 that although the ReLU activation performed better in the case of the ConFuse, however, in the case of DeConFuse, it was SELU that performed better. As the convolution layer was added, more resultant values from convolution with filters were believed to be negative. Since the ReLU activation function sets the negative values to zero, the values were not that distinct compared to those obtained via SELU. The latter does not set negative values to zero but near zero [96] and hence prevents dead neuron issue also unlike ReLU. It proved beneficial for performance, as shown by the numerical results in section 2.3.

2.3 Experimental Evaluation

Experiments were conducted for both frameworks on the real-world problems of stock forecasting and Trading. The problem of stock forecasting is a regression problem aiming at estimating the price of a stock at a future date (the next day for the given problem) given inputs till the current date. Stock trading is a

classification problem, where the decision to buy or sell a stock has to be taken at each time. The two problems are related by the fact that simple logic dictates that if the price of a stock at a later date is expected to increase, the stock must be bought, and if the stock price is expected to go down, the stock must be sold.

Five raw inputs were used for both tasks, namely open price, close price, high price, low price and net asset value (NAV). One could compute technical indicators based on the raw inputs [81] but, in keeping with the essence of true representation learning, it was deliberately chosen to stay with those raw values. Each of the five inputs was processed by a separate 1D processing pipeline. Each of the pipelines produced a flattened output. The flattened outputs were then concatenated and fed into the Transform Learning layer acting as the fully connected layer (Fig. 2.2) for fusion. While the processing pipeline ended here (unsupervised), the benchmark techniques were supervised and had an output node. The node was binary (buy/sell) for classification and real-valued for regression. The comparison with state-of-the-art time series analysis models, namely TimeNet [76] and ConvTimeNet [78] was carried out. In the former, the individual processing pipelines are based on LSTM and 1D CNN in the latter. The complete architectural details and hyperparameters for ConFuse and DeConFuse are in Table 2.1.

Table 2.1: **Description of compared models with hyperparameters**

Method	Architecture Description	Other Parameters
ConFuse	$5 \times$ { **layer1**: **1D Conv**$(1, 4, 5, 1, 2)$[1]; **Activation** (e.g., ReLU) } $1 \times$ **layer2**: **Transform Learning**	Learning Rate = 0.001, $\mu = 0.01, \lambda = 0.0001$ **Optimizer Used**: **Adam** **with parameters** $(\beta 1, \beta 2) = (0.9, 0.999)$, weight_decay = 5e-5, epsilon = 1e-8
DeConFuse	$5 \times$ { **layer1** : **1D Conv**$(1, 4, 5, 1, 2)$[1]; **Maxpool**$(2, 2)$[2]; **SELU**; **layer2** : **1D Conv**$(5, 8, 3, 1, 1)$[1] } **layer3** : **Fully Connected**	
ConvTimeNet	$5\times$ { **layer1** : **1D Convolution**$(1, 32, 9, 1, 4)$[1]; **Batch Normalization** + **SELU**; **layer2** : **1D Convolution**$(32, 32, 3, 1, 1)$[1]; **Batch Normalization** + **SELU** + **SC**[3]; **layer3** : **1D Convolution**$(32, 64, 9, 1, 4)$[1]; **Batch Normalization** + **SELU**; **layer4** : **1D Convolution**$(64, 64, 3, 1, 1)$[1]; **Batch Normalization** + **SELU** + **SC**[3]; **layer3** : **Global Average Pooling** } **layer4** : **Fully Connected** **For Trading, added layer5** : **Softmax**	**For Forecasting:** Learning Rate = 0.001, **For Trading:** Learning Rate = 0.0001, **Optimizer Used**: **Adam** **with parameters** $(\beta 1, \beta 2) = (0.9, 0.999)$, weight_decay = 1e-4, epsilon = 1e-8
TimeNet	$5\times$ { **layer1** : **LSTM unit**$(1, 12, 2, True)$[4]; **layer2** : **Global Average Pooling** } **layer3** : **Fully Connected** **For Trading, added layer4** : **Softmax**	**For Forecasting:** Learning Rate = 0.001, **For Trading:** Learning Rate = 0.0005, **Optimizer Used**: **Adam** **with parameters** $(\beta 1, \beta 2) = (0.9, 0.999)$, weight_decay = 5e-5, epsilon = 1e-8

[1] (in_planes, out_planes, kernel_size, stride, padding)
[2] (kernel_size, stride)
[3] SC - Skip-Connection
[4] (input_size,hidden_size,#layers,bidirectional)

2.4 Results and Analysis

The frameworks have been applied on the NSE dataset of 150 symbols, as also described in section 1.3.1. For DeConFuse, TimeNet and ConvTimeNet, the

Table 2.2: **Forecasting Results (MAE)**

Method	**Open**	**Close**	**High**	**Low**	**NAV**
ConFuse-SELU	0.011	0.023	0.017	0.017	0.447
ConFuse-ReLU	0.009	0.021	0.014	0.014	0.445
ConFuse-PReLU	0.007	0.017	0.012	0.013	0.434
ConFuse-LeakyReLU	**0.007**	**0.017**	**0.012**	**0.013**	**0.427**
ConFuse-Tanh	0.258	0.259	0.258	0.259	0.488
ConFuse-Sigmoid	0.227	0.227	0.227	0.227	0.482
ConvTimeNet	1.551	1.554	1.535	1.567	2.357
TimeNet	0.295	0.295	0.294	0.296	0.511

architectures were tuned to yield the best performance and randomly initialized the weights for each stock's training.

2.4.1 Stock Forecasting – Regression

Firstly, the experiments were performed with the stock forecasting problem. Next, the generated unsupervised features were fed from the proposed architecture into an external regressor - ridge regression. Evaluation was carried out regarding mean absolute error (MAE) between the predicted and actual stock prices for all 150 stocks. Root Mean Squared Error(RMSE) could also have been computed in place of MAE but MAE was chosen here as MAE has lower sample variance and is more interpretable than RMSE. The MAE for individual stocks is computed for each of close price, open price, high price, low price and net asset value.

Results Analysis for ConFuse

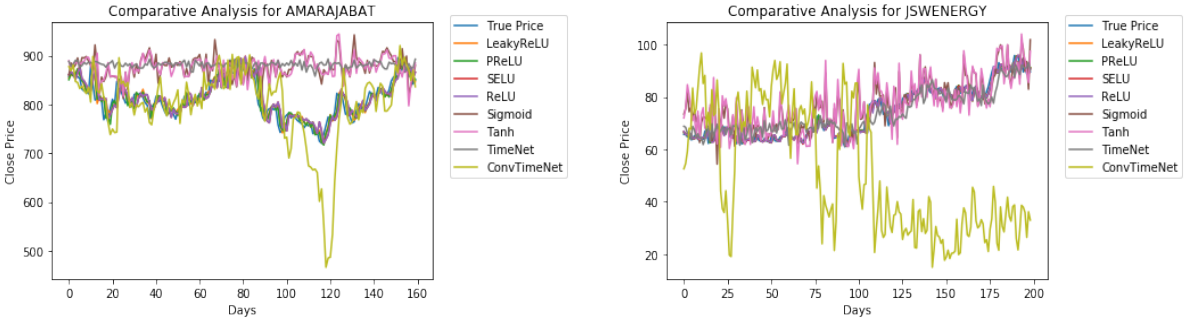
The testing was done with six different activation functions for *ConFuse*. [4] For a concise summary of results, Table 2.3 shows the average values over all stocks.

Table 2.3: **Summary Forecasting Results (MAE) with ConFuse**

Method	**Open**	**Close**	**High**	**Low**	**NAV**
ConFuse-SELU	0.011	0.023	0.017	0.017	0.447
ConFuse-ReLU	0.009	0.021	0.014	0.014	0.445
ConFuse-PReLU	0.007	0.017	0.012	0.013	0.434
ConFuse-LeakyReLU	**0.007**	**0.017**	**0.012**	**0.013**	**0.427**
ConFuse-Tanh	0.258	0.259	0.258	0.259	0.488
ConFuse-Sigmoid	0.227	0.227	0.227	0.227	0.482
ConvTimeNet	1.551	1.554	1.535	1.567	2.357
TimeNet	0.295	0.295	0.294	0.296	0.511

It was found that the results for the stock forecasting problem were exceptionally good. For most tested activation functions, *ConFuse* has MAE more than one order of magnitude lower than the state-of-the-arts. The regression performance was also plotted in the Figure 2.3 for the two randomly chosen stocks. Here, it was clearly observed that the output close prices were very closely predicted to the actual close prices versus the benchmarks.

[4] The gradients of SELU, RELU, PRELU, and Leaky RELU are not defined in zero. It is customary to consider any valid sub-gradient value instead. When resorting to this strategy, no practical convergence issues with the ADAM algorithm were found.



(a) AMARAJABAT (b) JSWENERGY

Figure 2.3: Stock Forecasting Performance with ConFuse

Results Analysis for DeConFuse

Summary results are presented in Table 2.4. Interested readers can see the detailed results for all 150 stocks from the paper [69], Appendix section. Table 2.4 shows that the MAE values reached for the proposed DeConFuse solution for the four first prices (open, close, high, low) are extremely good for all of the 150 stocks. Regarding NAV prediction, the proposed method performed extremely well for 128 stocks. For the remaining 22 stocks, there are 13 stocks, highlighted in red, for which DeConFuse did not give the lowest MAE, but it was still very close to the best results given by the TimeNet approach.

Table 2.4: **Summary Forecasting Results (MAE) with DeConFuse**

Method	**Open**	**Close**	**High**	**Low**	**NAV**
DeConFuse	**0.007**	**0.016**	**0.012**	**0.013**	**0.410**
ConvTimeNet	1.550	1.550	1.530	1.560	2.350
TimeNet	0.295	0.295	0.294	0.295	0.511

It can be observed that with both shallow (ConFuse) and Deep (DeConFuse) versions of the proposed frameworks, the forecasting performance is better than

the stat-of-the-arts. Further, going deep did better than the shallow version for a few predicted prices.

2.4.2 Stock Trading – Classification

Next, the Stock Trading, i.e., classification performance was evaluated. For this purpose, the features/representations Z were fed to the external classifier - Random Decision Forest (RDF). The results were reported in terms of metrics - precision, recall, F1 score, and area under the ROC curve (AUC). From the financial viewpoint, annualized returns (AR) were also calculated using the predicted trading signals/labels and true trading signals/labels named Predicted AR and True AR respectively. The latter metric is important from the perspective of understanding the quality of predictions in financial terms as well. This value indicates the geometric average of an investment's earnings in a year. Thus, the more it is, better is the quality of predictions. To calculate the same, the starting capital used for every stock was Rs. 1,00,000 and the transaction charges were Rs 10. Each of these metrics is explained below :

- **Accuracy** : the fraction of total samples that are correctly classified.

$$Accuracy = \frac{\#correct\,identified\,samples}{Total\#samples} \tag{2.16}$$

i.e.

$$Accuracy = \frac{\sum_{i=1}^{m} \frac{TP_i+TN_i}{TP_i+TN_i+FN_i+FP_i}}{m} \tag{2.17}$$

where TP_i = True Positives, TN_i = True Negatives, FP_i = False Positives,

FN_i = False Negatives, m is the total number of classes in the dataset, and i ranges from 1 to m.

- **Precision** : also known as the positive predictive value (PPV), measures the accuracy of a predicted positive outcome, i.e., how accurate the model is for predicting positive values.

$$Precision\, or\, PPV = \frac{TP}{TP + FP} \tag{2.18}$$

- **Recall** : represents the sensitivity of a model and is useful for ascertaining the strength of a model to predict positive outcomes.

$$Recall\, or\, Sensitivity = \frac{TP}{TP + FN} \tag{2.19}$$

- **F1 Score** : calculated using a weighted harmonic mean between precision and recall. For the classification of positive instances, it helps to understand the trade off between correctness and coverage.

$$F_\beta = (1 + \beta^2) * \frac{Precision \times Recall}{(\beta^2 \times Precision) + Recall} \tag{2.20}$$

here $\beta = 1$

- **ROC AUC** : ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters: True Positive Rate (TPR) and False Positive Rate (FPR). Here TPR is a synonym for recall and is therefore defined same as Recall mathematically.

False Positive Rate (FPR) is defined as follows:

$$FPR = \frac{FP}{FP + TN} \tag{2.21}$$

AUC stands for “Area under the ROC Curve." That is, AUC measures the entire two-dimensional area underneath the entire ROC curve.

- **AR** : indicates the geometric average of an investment’s earnings in a year.

$$AR = \left(\frac{TotalMoney}{StartCapital}\right)^{\frac{1}{n}} - Transactioncharges. \tag{2.22}$$

Here, transaction charges = Rs. 10/- and Start Capital = Rs. 1,00,000/-

Results Analysis for ConFuse

The results can be referred from Table 2.5. For the stock trading problem, *ConFuse* outperformed the benchmarks with two activation functions, namely SELU and ReLU, and reached a similar performance to the benchmarks with the other activation functions.

Table 2.5: **Trading Results with ConFuse**

Method	**Precis.**	**Recall**	**F1**	**AUC**	**AR**
ConFuse-SELU	**0.524**	**0.777**	**0.619**	**0.543**	**17.898**
ConFuse-RELU	0.505	0.648	0.556	0.523	18.112
ConFuse-PRELU	0.491	0.601	0.528	0.506	19.091
ConFuse-LeakyRELU	0.496	0.602	0.531	0.511	19.150
ConFuse-Tanh	0.469	0.560	0.493	0.497	19.002
ConFuse-Sigmoid	0.487	0.584	0.513	0.498	20.540
ConvTimeNet	0.457	0.507	0.413	0.524	19.410
TimeNet	0.469	0.648	0.496	0.513	18.764

Results Analysis for DeConFuse

The classification performance in detail for all 150 symbols can be referred from the paper Appendix Section. Certain results from that table are highlighted in bold or red. The first set of results, marked in bold, are the ones where one of the techniques for each metric gave the best performance for each stock. The proposed solution DeConFuse gave the best results for 89 stocks for a precision score, 85 stocks for a recall score, 125 stocks for F1 score, 91 stocks for the AUC measure, and 56 stocks in the case of the AR metric.

The other set marked in red highlighted the cases where DeConfuse did not perform the best but performed nearly equal (here, a difference of a maximum of 0.05 in the metric is considered) to the best performance given by one of the benchmarks, i.e., DeConFuse gave the next best performance. It was noticed that there are 24 stocks for which DeConFuse gave the next best precision metric

value. Likewise, 18 stocks in case of a recall, 22 stocks for F1 score, 26 stocks for AUC values, and 1 stock in case of AR. Overall, DeConFuse reached a very satisfying performance over the benchmark techniques. The trading results summary corroborates the same in Table 2.6.

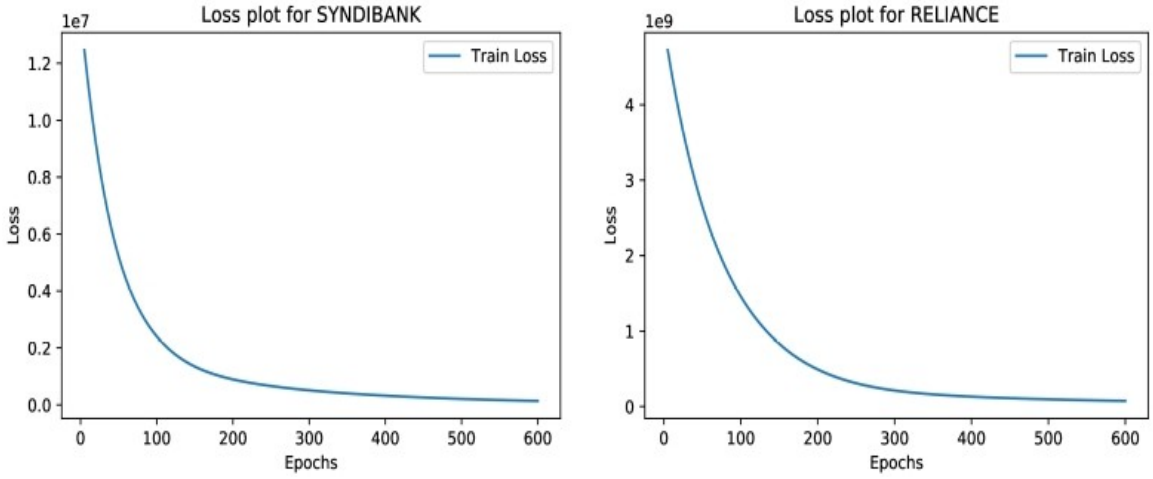
Table 2.6: **Summary Trading Results with DeConFuse**

Method	**Precision**	**Recall**	**F1 Score**	**AUC**	**MAE AR**
DeConFuse	**0.520**	**0.810**	**0.628**	**0.543**	**17.350**
ConvTimeNet	0.510	0.457	0.413	0.524	19.410
TimeNet	0.470	0.648	0.490	0.513	18.760

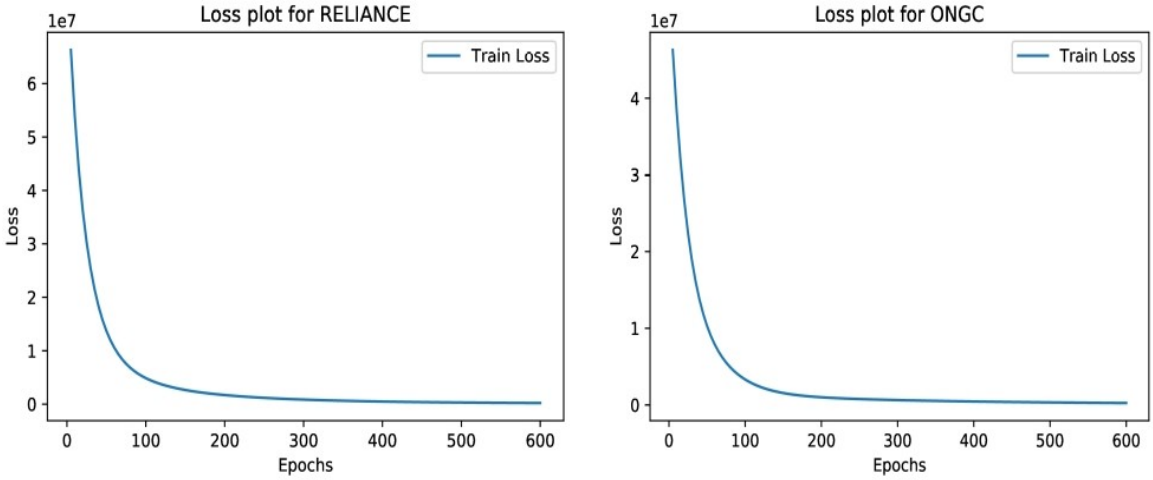
2.4.3 Convergence Study

Some empirical convergence plots of Adam were also shown that can be seen in Figure 2.4 when using *ConFuse* and *DeConFuse* with SELU, which depicted the practical stability of the end-to-end training method.

Further, the representations both channel-wise i.e., X_c and final fused representation Z, were analyzed for one of the random stocks; here it is ANDHRABANK. The visualizations are displayed in Figure 2.5 for one sample of the mentioned stock. It can be seen from the figure that the heatmaps for all the channel-wise features X_c and fused features Z are less redundant and have more variations. Thus, it can be implied that one of the factors for this variation could be distinct filters that are learned and transform data to produce the varied representations.

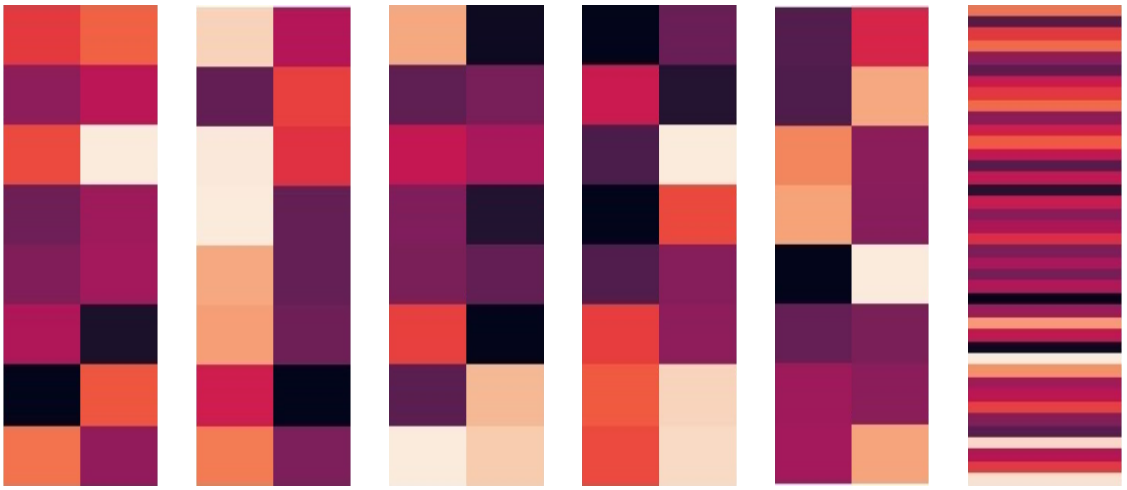


(a) Loss Plot with *ConFuse*



(b) Loss Plot with *DeConFuse*

Figure 2.4: Loss Plots with a) ConFuse and b) DeConFuse



(a) Channel X_1 Close Price | (b) Channel X_2 Open Price | (c) Channel X_3 High Price | (d) Channel X_4 Low Price | (e) Channel X_5 Net Asset Value | (f) Channel Z - Fused features

Figure 2.5: Visualization of channel-wise features X_c and fused representations Z for DeConFuse for one sample of stock ANDHRABANK (with 8×2 as the shape of the features obtained for each channel X_c and flattened features of shape 40×1 for Z)

2.5 Discussion

Shallow and deep fusion based end-to-end frameworks for processing 1D multi-channel data were proposed. Unlike other deep learning models, these frameworks are unsupervised. These are based on a novel deep version of the recently proposed CTL model. The proposed models have been applied for stock forecasting and trading problems leading to very good performance. The overall framework is generic enough to handle other multi-channel fusion problems as well.

The advantage of the proposed frameworks is their ability to learn in an unsupervised fashion. For example, consider the problem that is addressed. For traditional deep learning-based models, one needs to re-train deep networks for regression and classification. But here the learned final features can be reused, without the requirement of re-training, for specific tasks. This has advantages in other areas as well. For example, one can either do ischemia detection, i.e., detect whether one is having a stroke at the current time instant (from EEG); or one can do ischemia prediction, i.e., forecast if a stroke is going to happen. In standard deep learning, two networks need to be re-trained and tuned to tackle these two problems. With these proposed methods, there is no need for this double effort.

Since the stock data is quite volatile, therefore, a minor improvement matters in the problems pertaining to this domain. Thus, the better results with the proposed frameworks than the benchmarks is beneficial for the system. However,

the AUC ROC values can be improved further in the future as those have the scope of improvement in this case of two classes problem. Also, in the the future, the framework can be extended for semi-supervised formulations. It is believed that the semi-supervised formulation will be of immense practical importance.

Chapter 3

Supervised multi-channel fusion frameworks - SuperDeConFuse and DeConDFFuse

In the last chapter, the unsupervised frameworks based on CTL were discussed that bridged all the gaps that CNNs have. However, the question that comes next is - if we have labeled datasets, are the CNNs based models sufficient for supervised learning? It has been observed that CNNs have emerged as the recommended solution in many such scenarios. But the issue with CNN is that the supervised learning through them does not ensure distinct filters; hence, the feature maps might have redundancy. Additionally, there is a dead neuron problem with CNNs which is encountered with the kind of activation function chosen with it and mostly happens when ReLU is used. A dead neuron can be considered a natural Dropout. Further, due to dead neurons, there could be a bigger problem. Let's say if every neuron in a specific hidden layer is dead; it

cuts the gradient to the previous layer resulting in zero gradients to the layers behind it. Thus, the weights would not be updated and the learning will be improper. It can be fixed using lower learning rates, so the big gradient doesn't set a big negative weight and bias in a ReLU neuron. Another solution is to use other activation functions like Leaky ReLU. It allows the neurons outside the active interval to leak some gradient backward. But sometimes, the resolves just discussed do not work in certain scenarios.

Therefore, these two issues discussed above open up the scope for developing supervised frameworks that can combinely tackle them. In the previous chapter, the success of unsupervised frameworks based on CTL was observed. This chapter also presents two multi-channel supervised frameworks based on CTL. The first one is SuperDeConFuse, a multi-channel fusion framework that jointly trains and optimizes multiple CTL based channels and cross-entropy loss. Thus, representations are not learned just via CTL but also directed by classification loss - Cross Entropy. It has been applied to the stock trading problem. The other framework is named - DeConDFFuse combines and jointly trains DeConFuse and Decision Forest (DF). It deals with the drug-drug interaction problem. Here, the representations are learned via CTL and DF, which yields better performance than the state-of-the-arts. Both frameworks are explained in the subsequent sections.

3.1 SuperDeConFuse: A supervised deep convolutional transform based fusion framework for financial trading systems

3.1.1 Literature Review - Stock Trading

Information Fusion based techniques, in general, have been discussed in chapter 1. Now, let's briefly review here some of the works that have proposed solutions for the Stock Trading problem. The problem of stock trading has been one of the most difficult problems for researchers in finance data processing and speculators. Struggles are mainly due to the uncertainties and noises of the samples. These samples are generated as a consequence of historical market behaviors.

In literature, different methodologies have been applied to the stock data for predicting future trading strategies (e.g., buy and sell decisions). These include statistical methods, machine learning algorithms like Support Vector Machine (SVM) and Artificial Neural Networks (ANN), feature extraction approaches, deep learning models (e.g., CNN, LSTM) and self-supervised learning based techniques that are briefly reviewed in this section.

Statistical methods are probably the methods that are universally used for predicting financial stock trading strategies. In particular, many studies rely on the use of sequential statistical models, such as ARMA [100], ARCH [101], GARCH [102] and [103], Kalman filter [104]. Feature-based techniques are also considered state-of-the-art. Technical indicators like Exponential moving average (EMA), Moving average convergence and divergence (MACD), Williams %R,

etc., have been used in past studies to extract the features from the data [105]. Text mining can also be used to process financial analysis from newspapers [106]. The features are then input to machine learning models, for example, SVM, ANN and kNN [107].

Further studies have proposed hybrid machine learning models using multiple base classifiers operating on a common input and a meta classifier learning from base classifiers' outputs to obtain more precise stock return and risk predictions. Strategies such as Bagging, Boosting and AdaBoost can be applied to create diversity in classifier combinations [108, 109]. For example, a hybrid weighted SVM and weighted KNN model for predicting stock market indices is proposed in [110]. Another study [111] combines the statistical and probabilistic Bayesian Learning and the machine learning model ANN for the same. However, in all the aforementioned techniques, the relationship built between historical data and future value prediction may lack interpretation because of their "black-box" property. Thus, the performance of these methods is directly related to the quality of the features. Moreover, overfitting is a major issue with machine learning techniques due to their non-linear mapping and fitting capability.

Deep learning based models have also been extensively used for solving stock forecasting problems. Recurrent Neural Networks (RNNs) are considered the most appropriate models for time-series analysis. LSTM is one such RNN that is regarded as the memory-mimicking model. The work in [112] uses LSTM on the technical indicators for the prediction. However, despite the great performance obtained, the time complexity of training RNN via backpropagation

has encouraged the users to search for more tractable models and solutions.

CNNs constitute another important deep learning model, besides RNNs, which have been used profusely and performed well in stock time-series forecasting, especially 2-D CNNs. The studies pertaining to CNNs [79–85] have been discussed in the previous chapter in section 2.1.1. It is observed that there is a lack of performance improvement may be owing to the incorrect choice of the CNN model since these studies model an inherently 1D time series as an image.

Another learning paradigm known as Self-Supervised Learning (SSL) based models are also emerging currently when no labels for the data are available. In all such techniques, initially the data is unsupervised which is eventually turned supervised by predicting the pseudo labels and then training happens. There are few works that utilizes and proposes solutions based on it for stock trading prediction [86–89]. However, such techniques are resource intense and just like CNNs, these SSL based learning paradigms do not have distinctiveness guarantees.

3.1.2 Proposed Formulation

A novel supervised framework for multi-channel data representation learning is discussed in this section. A crucial element of the latter is the recently introduced CTL [56]. The details of CTL have already been covered in 2.1.2. Also, extending it to the deep versions and the fusion part is covered with DeConFuse described in section 2.2.2. Now, let's move to the proposed framework which

is an extension of these approaches to handle a multi-layer architecture that is called as - SuperDeConFuse (SDCF)[1] architecture.

This framework took the channels of input data samples to separate branches of CTL layers, leading to multiple sets of channel-wise features. The features obtained were thus decoupled. In order to couple (i.e., fuse) them, these were concatenated and passed to a fully-connected layer, which yielded a set of distinct coupled features via transform learning. These features were then fed to another linear fully-connected layer. The obtained features that were finally inputted to the softmax layer which yielded probabilities for the classes. The complete architecture is shown in Figure 3.1.

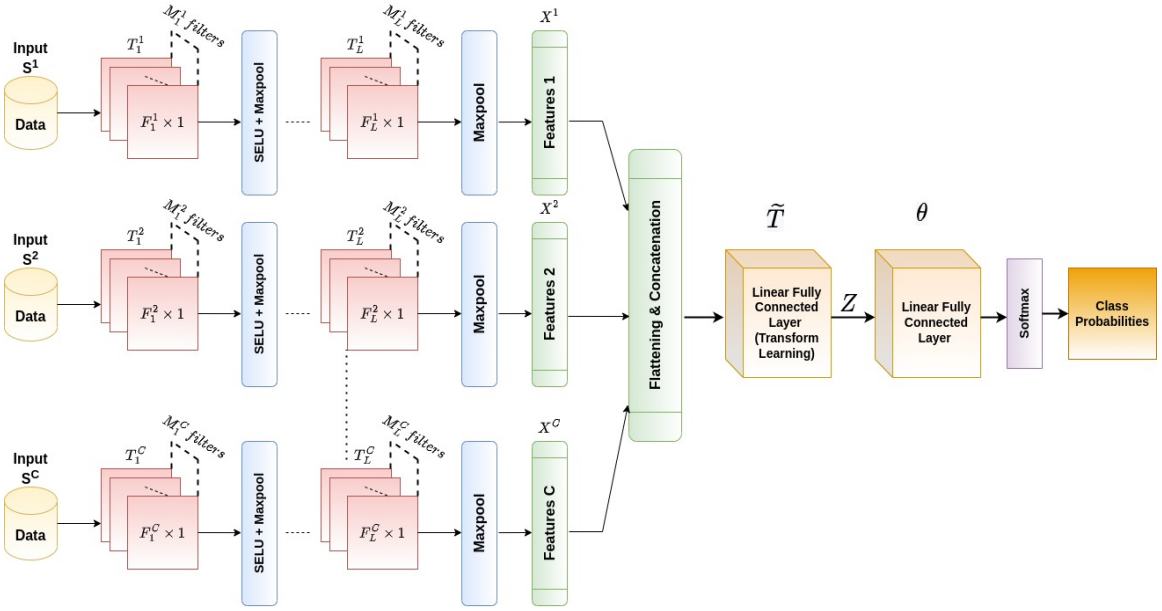


Figure 3.1: General SuperDeConfuse Architecture. The architecture is tested for $L = 1, 2, 3, 4$ layers and $C = 5$. Here $M_1^1 \times 1, \ldots, M_L^C \times 1$ represents the kernel size used in each layer $\ell \in \{1, \ldots, L\}$. Here, maxpooling is not performed after layer 4 due to the small window size/input sequence length.

As the data considered is multi-channel, a different set of convolutional filters

[1]P. Gupta, A. Majumdar, E. Chouzenoux, G. Chierchia, SuperDeConFuse: A supervised deep convolutional transform based fusion framework for financial trading systems, Expert Systems with Applications, Volume 169, 2021, 114206, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2020.114206

$T_1^{(c)}, \ldots, T_L^{(c)}$ and features $X^{(c)}$ were learned for each channel $c \in \{1, \ldots, C\}$. The linear transform (not convolutional) were also learned and calculated $\widetilde{T} = (\widetilde{T}_c)_{1 \leq c \leq C}$ to fuse the channel-wise features $X = (X^{(c)})_{1 \leq c \leq C}$, along with the corresponding fused features Z at the same time. The latter task was carried out by the cost function

$$F_{\text{fusion}}(\widetilde{T}, Z, X) = \frac{1}{2}\left\| Z - \sum_{c=1}^{C} \text{flat}(X^{(c)})\widetilde{T}_c \right\|_F^2 + \Psi(Z) + \sum_{c=1}^{C} \left(\mu \left\| \widetilde{T}_c \right\|_F^2 - \lambda \log\det(\widetilde{T}_c) \right) \tag{3.1}$$

where the operator "flat" transforms $X^{(c)}$ into a matrix where each row contains the "flattened" features of a sample. Further, the weight matrix θ of a multiclass classifier was learned which took the input features Z and yielded the class probabilities. The cross-entropy (CE) loss associated with the final classification is given by

$$F_{\text{CE}}(\theta, Z \mid y) = \sum_{k=1}^{K} \log \Big(\sum_{v=1}^{V} e^{z_k^\top (\theta_v - \theta_{y_k})} \Big), \tag{3.2}$$

where V is the number of classes, θ_v is the v-th column of matrix θ, $z_k^\top$ is the k-th row of matrix Z, and $y_k \in \{1, \ldots, V\}$ is the label of the k-th sample.

Conclusively, the proposed formulation aimed at jointly training the channel-wise convolutional filters $T_l^{(c)}$, the fusion coefficients $\widetilde{T}$, and the multiclass classifier θ in an end-to-end fashion. The features X and Z were explicitly learned subjected to the regularization Ψ so as to avoid the problem of dead neurons. Moreover, the "log-det" regularization on both $T_l^{(c)}$ and $\widetilde{T}$ broke the

symmetry and enforced the diversity in the learned transforms. In contrast, the Frobenius regularization kept the transform coefficients bounded.

3.1.3 Optimization algorithm

It was chosen to find a local minimizer to the non-convex Problem (2.10) through the projected (sub)gradient descent, whose iterations read:

$$
\begin{aligned}
&\text{For } n = 0, 1, ... \\
&\left\lfloor
\begin{aligned}
T^{[n+1]} &= T^{[n]} - \gamma \nabla_T J(T^{[n]}, X^{[n]}, \widetilde{T}^{[n]}, Z^{[n]}, \theta^{[n]}) \\
X^{[n+1]} &= \mathcal{P}_+(X^{[n]} - \gamma \nabla_X J(T^{[n]}, X^{[n]}, \widetilde{T}^{[n]}, Z^{[n]}, \theta^{[n]})) \\
\widetilde{T}^{[n+1]} &= \widetilde{T}^{[n]} - \gamma \nabla_{\widetilde{T}} J(T^{[n]}, X^{[n]}, \widetilde{T}^{[n]}, Z^{[n]}, \theta^{[n]}) \\
Z^{[n+1]} &= \mathcal{P}_+(Z^{[n]} - \gamma \nabla_Z J(T^{[n]}, X^{[n]}, \widetilde{T}^{[n]}, Z^{[n]}, \theta^{[n]})) \\
\theta^{[n+1]} &= \theta^{[n]} - \gamma \nabla_\theta J(T^{[n]}, X^{[n]}, \widetilde{T}^{[n]}, Z^{[n]}, \theta^{[n]})
\end{aligned}
\right.
\end{aligned}
\tag{3.3}
$$

with $\mathcal{P}_+ = \max\{\cdot, 0\}$ (applied element-wise). It was initialized with some random matrices $T^{[0]}, X^{[0]}, \widetilde{T}^{[0]}, Z^{[0]}, \theta^{[0]}$ and a suitable step size $\gamma > 0$ was chosen. The gradient step was numerically evaluated with the accelerated scheme initially introduced for the ADAM method in [95]. The advantages of this optimization method have been discussed in the previous chapter 2.

3.1.4 Preprocessing

Before proceeding with the experimental setup, the labeling process for the dataset and training details are discussed here. The specifics of the dataset can be referred from the section 1.3.2.

3.1.4.1 Labeling Process

In the labeling phase, the labels were manually assigned to the daily close prices as Buy (0), Hold (1) and Sell (2). The labels were determined by performing a grid search on the list of holding percentages to identify the percentage change for which the stocks should be held to maximize the annualized returns for the company. Algorithm 1 gave the details of the labeling process.

3.1.4.2 Training Details

In general, the sliding walk forward validation technique is used as the cross-validation technique in the case of time-series data, also shown in Figure 3.2. As can be seen from Figure 3.2, ten years of data for training have been used and the subsequent one year of data for testing, i.e., the stock data from 1998-2007 was for training and the year 2008 for testing. Then the training window was slid by one year which implied that it was next trained from 1999-2008 and tested on the following year 2009 data and this period is called the horizon. In summary, it was trained for ten years, tested for the next year, slid it by a one year horizon, and again trained and tested it until 2018. Thus, 11 years

Algorithm 1: Labelling Method

Input : CP - Array of
Parameter : X - array of K holding percentages,
NUMDAYS - number of days for the current symbol or len(CP)
Labels - 2D array of size K x NUMDAYS
Output : FinalLabels - Labelled Dataset for S

AR = [ ] //it is of size K
for $k = 0, 1, 2, \ldots, K-1$ **do**
  for $n = 0, \ldots, NUMDAYS - 1$ **do**
    change = abs($(CP[n+1] - CP[n]/CP[n]) * 100$) //where CP[n+1] is the next day closing price
    if change $>$ X[k] **then**
      if $CP[n+1] > CP[n]$ **then**
        label == "Sell"
      else
        label == "Buy"
      end if
    else
      label == "Hold"
    end if
    Labels[k].append(label)
  end for
  ar = AnnualisedReturn(Labels[k],CP)
  AR.append(ar)
end for
maxAr = Max(AR), maxIndex = index(Max(AR))
HoldPercentage = X[maxIndex]
FinalLabels = Labels[maxIndex]
return FinalLabels
Repeat all steps till 22 for all the Stocks/Symbols in the dataset.

of data from 2008 - 2018 were used as test data. This way, there were 11 models and the set of hyperparameters were selected that gave the best results across all 11 models. The set of hyperparameters that were tuned includes μ, λ, kernel sizes, number of filters/kernels, learning rate, weight decay of the Adam optimizer, batch size, and number of epochs. Additionally, the weights for each stock's training were randomly initialized. It appeared here as a very efficient technique to analyze the robustness of the architecture. In other words, the model performance was calculated every time a year's data became available for testing

and used the previous year's test data for training. The training and the test data were standardized using Normalizer from the Python library as prices and the NAV features/channels have a varied range of values.

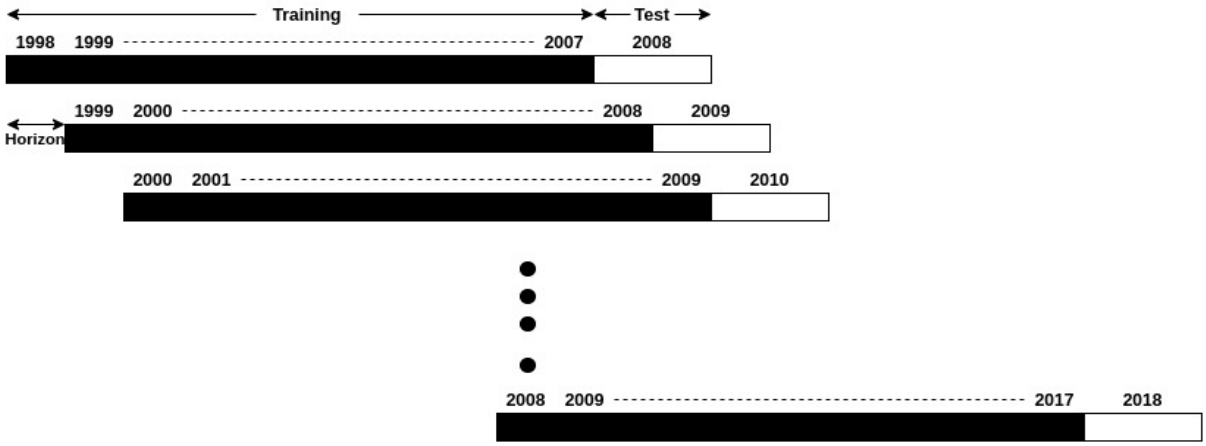


Figure 3.2: Sliding walk-forward validation technique used for hyperparameters tuning

3.1.5 Experimental Evaluation

The experiments were carried out on the real-world problem of stock trading. Stock trading is a classification problem, where the decision whether to buy or hold or sell a stock has to be taken at each time. The problem makes a decision that if the price of a stock at a later date is expected to increase, the stock must be bought; and if the stock price is expected to go down, the stock must be sold; and if there is no change in the price then it should be held, i.e., do nothing until the price increases. This was done in a way to maximize the annualized returns from the stock for the company's profit, as mentioned in the labeling process.

Five raw inputs were used: open price, close price, high, low and net asset value (NAV). It was chosen to stay with the raw values. However, one could compute technical indicators based on the raw inputs [105] but raw values

allowed here to keep up with the essence of the true nature of representation learning. Each of the five inputs was processed by a separate 1D processing pipeline. Each pipeline produced a flattened output (Figure 3.1). These flattened outputs were then concatenated and fed for fusion into the Transform Learning layer acting as the fully connected layer (Figure 3.1). Further, this is connected to another linear fully connected layer and finally, there was a softmax function. The softmax function gave the classification output which consisted of the class probabilities for the three classes (BUY, HOLD and SELL).

The architecture was extended by adding CTL layers upto four layers resulting in four different deep SDCF architectures. The details for all four architectures are briefed in Table 3.1. Maxpooling halves the input sequence length/window size/Time Steps with every operation. Thus, after three layers, the size was getting reduced to the value that restricted us from using maxpooling operation after the 4^{th} CTL layer; hence, the architecture with 4 CTL layers of SDCF will not have maxpooling operation after layer 4. This was due to the small window size. Also, for making predictions on any day, the past ten days were analyzed through the model labeled as Time Steps shown in Figure 3.1. Additionally, the stock trading signal was not predicted for the first ten days of every test year to avoid data leak.

Table 3.1: **Hyperparameters for the different instances of the proposed SDCF network (see Figure 3.1 for the general overview) used in the experimental section.**

Method	Architecture Description	Other Parameters
SDCF 1L	$5 \times \begin{cases} \textbf{layer1} : \textbf{1D Conv}(1, 16, 3, 1, 1)^1 \\ \textbf{Maxpool}(2, 2)^2 \end{cases}$ **layer2** : **Fully Connected (TL)^3** **layer3** : **Fully Connected (Linear)** **Softmax**	$LearningRate = 0.001$, $\lambda = 0.01, \mu = 0.0001$ $epochs = 100$, **Optimizer Used**: **Adam** **with parameters** $(\beta 1, \beta 2) = (0.9, 0.999)$, weight_decay = 1e-4, epsilon = 1e-8
SDCF 2L	$5 \times \begin{cases} \textbf{layer1} : \textbf{1D Conv}(1, 8, 3, 1, 1)^1 \\ \textbf{SELU} + \textbf{Maxpool}(2, 2)^2 \\ \textbf{layer2} : \textbf{1D Conv}(8, 16, 3, 1, 1)^1 \\ \textbf{Maxpool}(2, 2)^2 \end{cases}$ **layer3** : **Fully Connected (TL)^3** **layer4** : **Fully Connected (Linear)** **Softmax**	
SDCF 3L	$5 \times \begin{cases} \textbf{layer1} : \textbf{1D Conv}(1, 4, 11, 1, 5)^1 \\ \textbf{SELU} + \textbf{Maxpool}(2, 2)^2 \\ \textbf{layer2} : \textbf{1D Conv}(4, 8, 7, 1, 3)^1 \\ \textbf{SELU} + \textbf{Maxpool}(2, 2)^2 \\ \textbf{layer3} : \textbf{1D Conv}(8, 16, 3, 1, 1)^1 \\ \textbf{Maxpool}(2, 2)^2 \end{cases}$ **layer4** : **Fully Connected (TL)^3** **layer5** : **Fully Connected (Linear)** **Softmax**	

SDCF 4L	$5 \times$ { **layer1** : **1D Conv**$(1, 4, 13, 1, 6)^1$ **SELU** + **Maxpool**$(2, 2)^2$ **layer2** : **1D Conv**$(4, 8, 11, 1, 5)^1$ **SELU** + **Maxpool**$(2, 2)^2$ **layer3** : **1D Conv**$(8, 16, 9, 1, 4)^1$ **SELU** + **Maxpool**$(2, 2)^2$ **layer4** : **1D Conv**$(16, 32, 5, 1, 2)^1$ } **layer5** : **Fully Connected (TL)**[3] **layer6** : **Fully Connected (Linear)** **Softmax**	

[1] (in_planes, out_planes, kernel_size, stride, padding)

[2] (kernel_size, stride)

[3] TL - Transform Learning

L - #CTL layers

The comparison was made with three state-of-the-art time series based analysis models, out of which two techniques presented the models proposed specifically for financial stock trading - CNN-TA [105] and MFNN [113]; and the last technique presented a generic model for time-series based data - FCN (Fully Convolutional Network) [75]. The latter was used to understand how generic the proposed model was when compared against both specific stock trading based and general time-series models. In all the techniques, processing pipelines were based on CNN. Other than CNN, MFNN [113] was also based on the RNN type of network - LSTM. In [105], the data used was not raw but processed as technical indicator values and passed as an image, hence using 2D CNN, whereas, in FCN [75], the data was processed via 1D CNN. The same hyperparameters for

the benchmark techniques were used as given in the study, except for FCN which was best tuned for the used dataset. It was also compared to the simple CNN with the architecture same as that of proposed framework, i.e., 3 convolutional layers deep architecture and used the same hyperparameters too, except the kernel sizes of $F_1 = 11$, $F_2 = 9$ and $F_3 = 7$ for the convolutional layers $\ell = 1, 2$ and 3 (padding size is $F_\ell/2$). The difference lied in the objective function of the convolutional learning in both the techniques, i.e., 3 layers deep SDCF and 3 layers deep and simple 1D CNN. This was done to analyze the performance difference between the two supervised learning techniques. Additionally, the architecture for CNN was having 3 convolutional layers since the results were best with 3 convolutional layers and depleted after that.

3.1.6 Results and Analysis

The predictions from every year totaling 11 years were saved, and the metrics were computed to analyze the performance of the SDCF model. Two sets of metrics were computed here, namely (i) classification metrics and (ii) financial metrics.

(i) Classification Metrics - this set of metrics includes class-wise F1 score, Precision and Recall to assess the performance from a classification point of view. Also, the weighted F1 Score, Precision and Recall to account for the class imbalance for every stock were calculated. Note that, in such a case, the F1 score is not equivalent to the harmonic mean of Precision and

Recall since it is weighted.

(ii) Financial Metrics - Additionally, the evaluation of the performance of the proposed framework and state-of-the-art was carried out from the financial point of view. For the same purpose, Annualized Returns(AR) were computed using the predictions from all the models. The AR value was calculated the same way as mentioned in [105]. The starting capital was Rs 10,00,00,000.0, and transaction charges were Rs 10. Indian currency was used to calculate the AR values since the dataset had all Indian stocks. Note, however, that the chosen metric was versatile and could be used to evaluate the model in any currency depending on the stocks analyzed.

3.1.6.1 Classification Analysis

As mentioned previously, let's first look at the Classification performance of the proposed models. The framework was tested for shallow - 1 CTL layer and deeper versions - 2, 3 and 4 CTL layers. The generated features from the fully connected layers are passed to the softmax after which the probabilities for all the classes were obtained. The one with the maximum probability was selected as the predicted label. The performance was calculated for every class. Specifically, F1 Score, Precision and Recall metrics are computed for BUY, HOLD and SELL classes. Here, the summary results for each of the classes - BUY, SELL and HOLD signals to understand class wise results are given in the Tables 3.2, 3.4 and 3.3 and the global results from 3.5. The detailed results can be referred to from the tables in Appendix section .

Certain results are highlighted in bold or red. In the first set of results marked bold, one or more techniques for each metric give the best/greater than or equal performance. Analyzing it in detail, it is found that there are 8 stocks for which the proposed model performed greater than or equal to when compared with benchmark techniques for F1 score in the case of the BUY class. Following the same, it is found that the SDCF gave greater than or equal to performance for 13 stocks for precision and 5 stocks for recall metrics under the BUY class. Similarly, 7 stocks for the F1 score, 7 stocks for precision and 5 stocks for recall in the HOLD class and 7 stocks for the F1 score, 11 stocks for precision and 6 stocks for recall in the case of the SELL class. It was further analyzed to understand the performance difference between the supervised learning techniques, specifically, performance with CNN and the proposed model. It was found that CNN gave greater than or equal to performance for 2 stocks for each metric under the BUY class. Similarly, there are 6, 1 and 9 stocks for the HOLD class and 2 stocks each for the metrics F1 score, precision and recall under the SELL class.

Additionally, the other set of results in red indicates the performance where one of the proposed model versions gave the similar/next best performance under 0.02 error difference - err_dif (let's say) after one of the benchmarks, i.e., $0.0 <$ err_dif ≤ 0.02. Adhering to the same, it was observed that for the BUY class, there is 1 stock each for metrics F1 score, precision and recall, respectively. Likewise, for the HOLD class, there are 7, 4 and 5 stocks for F1 score, precision and recall metrics, respectively; and for the SELL class, 1 stock each for F1

score and recall metrics. Although the results for CNN haven't been highlighted when it gave similar/next-best performance but the statistics for the same are presented here. Analyzing for CNN, there are 2 and 3 stocks for F1 score and precision under the HOLD class. Observing these statistics, they indicate that the performance with the proposed model is better than CNN for all three BUY, HOLD and SELL classes.

Table 3.2: **Summary of BUY Class Classification Results for Stock Trading**

Method	**Avg. BUY F1 Score**	**Avg. BUY Precision**	**Avg. BUY Recall**
SDCF 1L	0.0645	0.2182	0.0475
SDCF 2L	0.0916	0.2356	0.0683
SDCF 3L	0.1091	0.2205	0.0854
SDCF 4L	**0.1566**	**0.3242**	0.1355
CNN	0.0688	0.1179	0.0551
FCN	0.0758	0.1446	0.0617
CNN-TA	0.1205	0.1611	0.1263
MFNN	0.0881	0.1672	**0.2401**

Table 3.3: **Summary of HOLD Class Classification Results for Stock Trading**

Method	**Avg. HOLD F1 Score**	**Avg. HOLD Precision**	**Avg. HOLD Recall**
SDCF 1L	**0.7983**	0.7091	**0.9446**
SDCF 2L	0.7912	0.7113	0.9164
SDCF 3L	0.7813	0.7113	0.8842
SDCF 4L	0.6684	0.5950	0.7960
CNN	0.7909	0.7090	0.9239
FCN	0.7825	0.7119	0.9051
CNN-TA	0.7686	**0.7142**	0.8557
MFNN	0.5161	0.6425	0.5718

Table 3.4: **Summary of SELL Class Classification Results for Stock Trading**

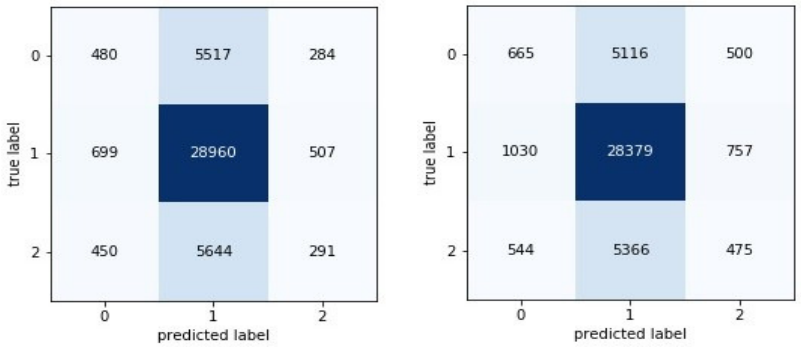
Method	**Avg. SELL F1 Score**	**Avg. SELL Precision**	**Avg. SELL Recall**
SDCF 1L	0.0423	0.1778	0.0285
SDCF 2L	0.0650	0.1752	0.0503
SDCF 3L	0.0759	0.1574	0.0635
SDCF 4L	**0.1410**	**0.2139**	0.1250
CNN	0.0481	0.0946	0.0379
FCN	0.0742	0.1658	0.0802
CNN-TA	0.0679	0.1768	0.0487
MFNN	0.0633	0.1034	**0.1734**

From the summary results in the above displayed tables, the average metric values for which the model gave the best performance are average F1 score and precision for the BUY class, average F1 score and recall for the HOLD class, and average F1 score and precision for the SELL class, where the F1 score is an

important metric, as it is the harmonic mean of precision and recall. It is the best with the proposed model - SDCF for all three classes.

As it can be observed, the performance for the HOLD class decreased when increasing the number of layers for the SDCF model. However, it can also be seen that there is an increase in correct identification for BUY and SELL points despite the fact that BUY and SELL points appear extremely less in the case of every stock as compared to HOLD points. The latter identification capacity is actually more crucial for the financial system as it directly influenced the financial gains or losses. Moreover, the overall individual class performance indicated that the model captured all three classes, i.e., BUY, HOLD and SELL well.

This was also indicated in the confusion matrices, given for each of the SDCF framework's shallow and deeper versions in Figure 3.3. With an increase in layers, the model started to identify the BUY and SELL points more correctly. The HOLD signal had more false positives with shallow architecture (SDCF 1L) that decreased with the increase in layer number, which was essential for the system in order to classify other class points correctly. Additionally, the overall performance of the proposed model was better than the CNN.



(a) CTL 1Layer

(b) CTL 2Layers

(c) CTL 3Layers

(d) CTL 4Layers

Figure 3.3: Confusion matrices corresponding to the different number of CTL layers of the architecture: a) 1 layer of CTL (shallow version), b) 2 layers of CTL (deep version), c) 3 layers of CTL (deep version) and d) 4 layers of CTL (deep version) where 0 - BUY, 1 - HOLD, 2 - SELL signals.

To better analyze the framework performance, the weighted F1 score, precision and recall metric values were calculated for all the stocks under consideration. The reason for computing weighted values was to incorporate the class imbalance for every stock. The detailed results can be referred from the appendix section of the paper [57] and summary results are given in Table 3.5. Again, the results comprised two sets of values marked in bold or red with the same err_dif of 0.02. There are 6, 9, and 5 stocks concerning the metrics F1 score, precision and recall for which the model performed greater than or equal to the performance given by the state-of-the-arts. Also, there are 6, 3 and 6 stocks for the metrics F1 score, precision and recall, respectively, for which the model gave

the next best performance under 0.02 err_dif. Although the BUY and SELL classes' performance with the 4 CTL Layers deep architecture is better than the benchmarks compared against, the overall performance from the average weighted metric is suggestive of the good performance with the 3 layers deep architecture classification wise. This is also suggested by the financial results explained later.

Again analyzing explicitly for CNN, there are 4, 2 and 7 stocks with greater than or equal performance; and 3, 2 and 3 stocks under similar/next best performance for the F1 score, precision and recall metrics, respectively. As can be referenced from the statistics presented here, the proposed model is giving better results with greater than or equal and the next best/similar performances except for the number of stocks for recall metric are slightly more with CNN under greater than or equal to performance. However, the next best performance statistic for the recall metric is much better than CNN. Overall performance on average is good with the proposed model as compared to the benchmarks and CNN which can also be referred from Table 3.5. For a deeper understanding of the aforementioned statistics, please refer to Table from [57].

Table 3.5: **Summary of Weighted Classification Results for Stock Trading**

Method	**Avg. F1 Score**	**Avg. Precision**	**Avg. Recall**
SDCF 1L	0.6169	**0.6216**	**0.6941**
SDCF 2L	0.6229	0.6207	0.6867
SDCF 3L	**0.6250**	0.6146	0.6784
SDCF 4L	0.5345	0.5464	0.5890
CNN	0.6182	0.5907	0.6898
FCN	0.6090	0.6079	0.6725
CNN-TA	0.6148	0.6161	0.6575
MFNN	0.4162	0.5509	0.4676

3.1.6.2 Financial Analysis

It is imperative to analyze the performance from a financial perspective to understand the quality of predictions made by the SDCF model. For this, as explained earlier, the AR values were calculated with the predictions generated by each technique for every stock over 11 years. The AR values were also calculated with the True labels for every stock over the same period. Finally, the absolute difference/error between the AR values from Predictions and the AR values from True labels was computed. The absolute difference values were averaged for all the stocks yielding the so-called Mean Absolute Error. The detailed results are given in Table from the paper [57]. With the proposed model, 5 stocks have the best performance whereas with CNN-TA, there is 1 stock and 2 stocks under MFNN and FCN. Overall, the performance is good with the proposed model as also evident from the summary results in Table 3.6 where

there is a mean of the absolute difference/error(MAE) between the True AR and Predicted AR. Also, there are 3 stocks for which the proposed model gave an equal performance as the other benchmark techniques. Here, this set of results illustrated that, despite the higher capability of identifying the BUY and SELL points with 4 layers deep CTL, the AR values are better predicted with the 3 layers deep CTL framework.

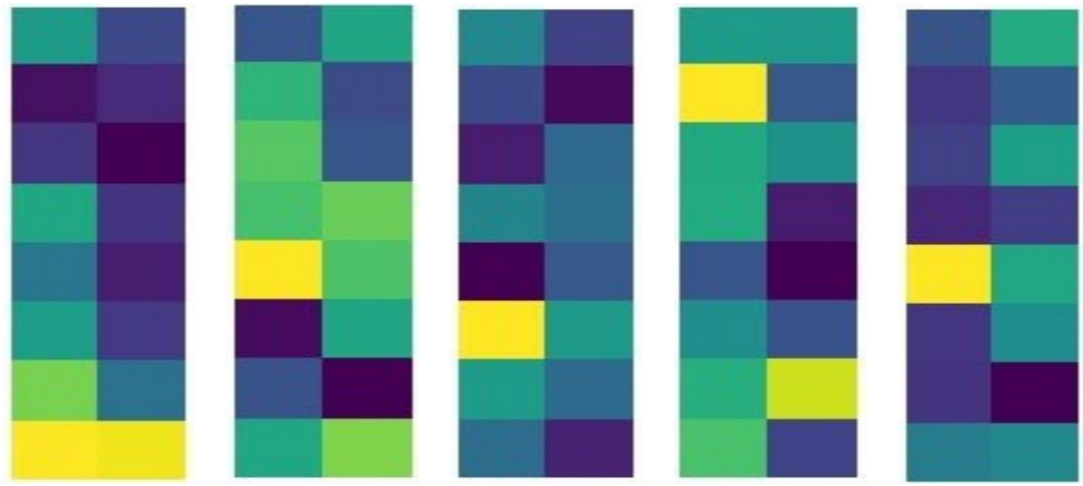
With respect to CNN, there are only 2 stocks for which CNN performs better than any benchmarks and the proposed models and 3 stocks for which it gave an equal performance. Thus, from the combined (greater than or equal to and next best / similar), average classification and financial results, the CNN results are less performant than the proposed model. This also indicated that the quality of predictions made with the SDCF model is better than CNN as the identified class labels give AR values quite close to the True AR values. This remained true for all the benchmarks. The statistics presented here can be deduced from Table in [57] for complete understanding.

Table 3.6: **Summary of Financial Results for Stock Trading**

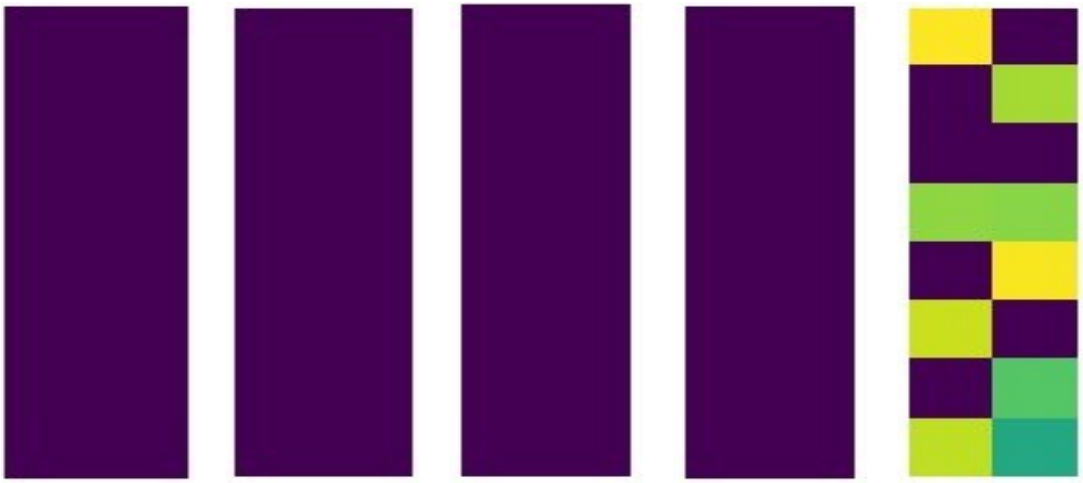
Method	**MAE AR**
SDCF 1L	22.5613
SDCF 2L	20.7227
SDCF 3L	**20.5067**
SDCF 4L	22.8287
CNN	21.1140
FCN	23.7720
CNN-TA	22.1380
MFNN	22.3040

To further understand the better supervised learning for both regular CNN and the SDCF framework, the channel-wise X_c features for both frameworks obtained after the last maxpool layer for the 3 convolutional layers deep framework were visualized. The following Figure 3.4 shows the visualizations of the features for one sample of the stock 'BSELINFRA.BO'.

Features generated by the proposed *SDCF* network.



Features generated by a standard CNN with a similar architecture.



(a) Channel X_1 Close Price (b) Channel X_2 Open Price (c) Channel X_3 High Price (d) Channel X_4 Low Price (e) Channel X_5 NAV

Figure 3.4: Visualization of channel-wise features X_c for SDCF versus a standard CNN for one sample of stock BSELINFRA.BO (with 16×1 as the shape of the features obtained and resized to 8×2 for better visualization)

As seen from Figure 3.4, the heatmap for each channel corresponding to the prices(Close, Open, High and Low) show no variation in the case of CNN compared to the SDCF architecture. While it shows some variations for the features learned corresponding to NAV, the features are still better learned with SDCF. Also, the darker the color in the heatmap, the more it is indicative of the larger negative exponent values. In the case of CNN, hence, the values are very very small that are almost diminishing to zero. This also corroborated the fact that the filters learned with the proposed model are distinct due to the "log-det" term added which further gives different features with significantly less redundancy. Thus, the visualizations of these channel-wise features are also supportive of better supervised training with the SDCF framework than CNN.

3.1.6.3 Ablation Studies

‘ In this section, the ablation study performed is discussed. The network was trained in a piecemeal fashion here. The motive for performing this study was to understand the behavior of the network without the benefit of joint training. Since it is piecemeal, it was carried out in two parts. In the first part, the network learned the representations Z in an unsupervised manner with the following objective function:

$$F_{\text{fusion}}(\widetilde{T}, Z, X) = \frac{1}{2}\left\| Z - \sum_{c=1}^{C} \text{flat}(X^{(c)})\widetilde{T}_c \right\|_F^2 + \Psi(Z) + \sum_{c=1}^{C} \left(\mu \left\| \widetilde{T}_c \right\|_F^2 - \lambda \log\det(\widetilde{T}_c) \right) \tag{3.4}$$

It is the same as present in equation 3.1 and hence all the variables here mean the same. Then these learned Z's are fed into the shallow single-layer neural network with the cross-entropy loss at the end separately, i.e.

$$F_{\text{CE}}(\theta, Z \mid y) = \sum_{k=1}^{K} \log \Big(\sum_{v=1}^{V} e^{z_k^\top (\theta_v - \theta_{y_k})} \Big), \tag{3.5}$$

It is also followed from the previously mentioned objective terms given by equation 3.2 and thus, all variables mean the same. However, the difference is that here θ is not learned jointly with Z, but two separate pipelines learn each of these variables individually. Also, the hyperparameters used for both parts are the same as the ones used in the proposed architecture with the Adam optimizer. The results corresponding to classification and financial analysis are given under heading **piecemeal** in tables 3.7, 3.8, 3.9, 3.10 and 3.11 respectively.

Table 3.7: **Ablation Study performance for BUY Class**

Method	**Avg. BUY F1 Score**	**Avg. BUY Precision**	**Avg. BUY Recall**
SDCF 1L	0.0645	0.2182	0.0475
SDCF 2L	0.0916	0.2356	0.0683
SDCF 3L	0.1091	0.2205	0.0854
SDCF 4L	**0.1566**	**0.3242**	**0.1355**
piecemeal	0.0449	0.1593	0.0379

Table 3.8: **Ablation Study performance for HOLD Class**

Method	**Avg. HOLD F1 Score**	**Avg. HOLD Precision**	**Avg. HOLD Recall**
SDCF 1L	0.7983	0.7091	0.9446
SDCF 2L	0.7912	0.7113	0.9164
SDCF 3L	0.7813	0.7113	0.8842
SDCF 4L	0.6684	0.5950	0.7960
piecemeal	**0.8002**	**0.7048**	**0.9592**

Table 3.9: **Ablation Study performance for SELL Class**

Method	**Avg. SELL F1 Score**	**Avg. SELL Precision**	**Avg. SELL Recall**
SDCF 1L	0.0423	0.1778	0.0285
SDCF 2L	0.0650	0.1752	0.0503
SDCF 3L	0.0759	0.1574	0.0635
SDCF 4L	**0.1410**	**0.2139**	**0.1250**
piecemeal	0.0221	0.1230	0.0127

Table 3.10: **Ablation Study performance weighted results**

Method	**Avg. F1 Score**	**Avg. Precision**	**Avg. Recall**
SDCF 1L	0.6169	**0.6216**	0.6941
SDCF 2L	0.6229	0.6207	0.6867
SDCF 3L	**0.6250**	0.6146	0.6784
SDCF 4L	0.5345	0.5464	0.5890
piecemeal	0.6090	0.6002	**0.6954**

Table 3.11: **Ablation Study Financial Results**

Method	MAE AR
SDCF 1L	22.5613
SDCF 2L	20.7227
SDCF 3L	**20.5067**
SDCF 4L	22.8287
piecemeal	23.4073

From the results in tables 3.7, 3.8, 3.9, 3.10 and 3.11, it can be clearly seen that the piecemeal version did not perform well as compared to the proposed solution except for the HOLD class and Recall value for weighted summary results under Classification performance category. The exception in results, however, have values only slightly better than the proposed ones. Despite the slightly higher results, the piecemeal approach did not recognize BUY and SELL points as efficiently as the proposed method - SDCF, which is critical for the system. It, thus, also suggests that the joint supervised solution involving cross-entropy loss guided the better representation learning. Therefore, the proposed solution's joint training is justified and important for the system to recognize critical points BUY and SELL as well as appropriately recognizing HOLD points efficiently that are comparable with other state-of-the-arts, CNNs, and piecemeal approaches.

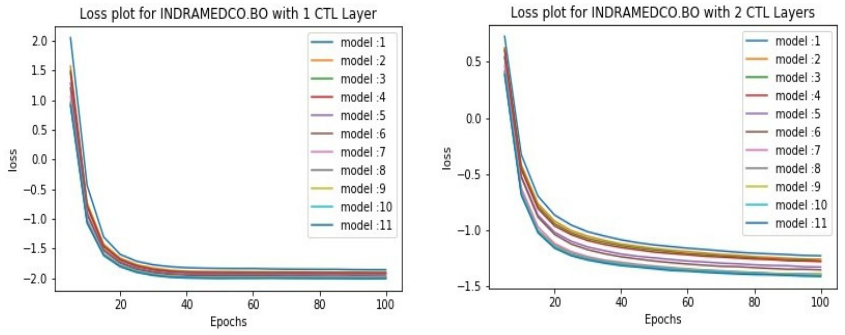
In order to test the proposed architecture's capability further, the experiments for two additional window sizes, namely 5 and 20 have been performed. In order to avoid extensive space utilization, only the comparative summary results are presented here - Weighted F1 Score(Classification Metric) and MAE

AR(Financial metric) in Table 3.12 for window sizes 5 and 20 along with the summarized results for window size 10. The proposed method yielded the best results on an aggregate. Even though CNN-TA yielded better AR for a solo case (window size 20), it did not reach better results in terms of weighted F1 for the same scenario. Furthermore, CNN-TA couldn't be run for all small window sizes (such as 5), hence cannot be deemed as an all-purpose go-to method. Small window sizes are crucial for highly non-stationary stocks and the inability of a technique to handle such stocks is a major shortcoming. Overall, the proposed model performed better than benchmarks and CNN both classification-wise and financially; specifically, it gave the best performance with 3 CTL layers deep SDCF framework of all the 4 SDCF architectures. The empirical convergence plots were also displayed for a few stocks, namely INDRAMEDCO.BO and NATIONALUM.BO in Figure 3.5 for both shallow and deeper versions. It can be seen that the training loss decreased to the point of stability for each example considered.

Table 3.12: **Comparative Summary Results for Stock Trading for window sizes 5,10,20**

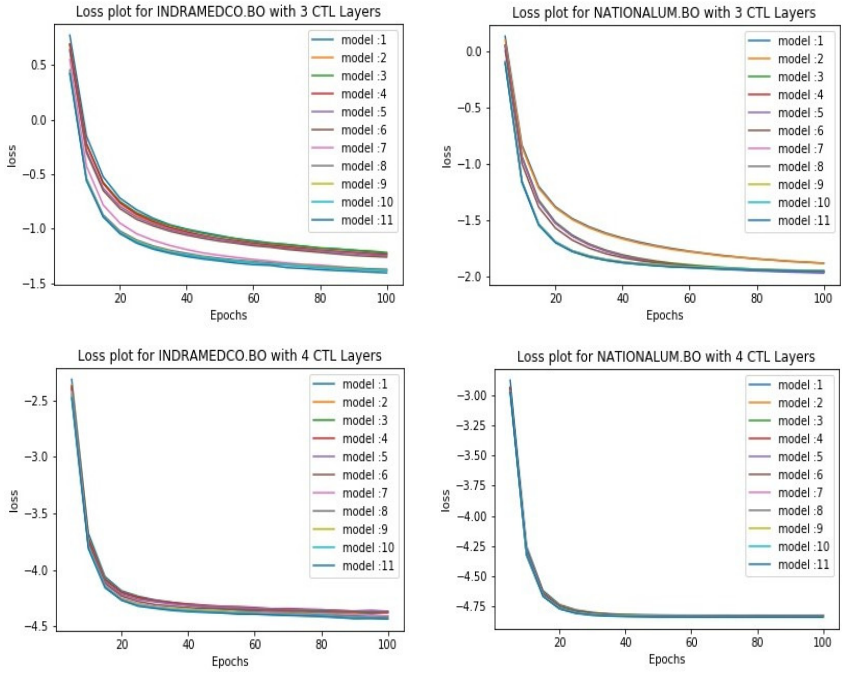
Method	**Window Size 5**		**Window Size 10**		**Window Size 20**	
	F1	**MAE AR**	**F1**	**MAE AR**	**F1**	**MAE AR**
SDCF 1L	0.6141	22.4947	0.6169	22.5613	0.6194	22.4453
SDCF 2L	0.6148	24.3820	0.6229	20.7227	0.6242	25.0200
SDCF 3L	**0.6207**	**20.9193**	**0.6250**	**20.5067**	**0.6262**	25.7667
SDCF 4L	0.6157	21.5427	0.5345	22.8287	0.6254	26.1007
CNN	0.6095	22.0113	0.6182	21.1140	0.6217	22.9560
FCN	0.6131	23.3107	0.6090	23.7720	0.6120	24.2233
CNN-TA*	-	-	0.6148	22.1380	0.6246	**20.3820**
MFNN	0.4105	23.4820	0.4162	22.3040	0.4869	23.2620

*CNN-TA cannot be run for window size 5 due to its inherent structure



(a) CTL 1 layer

(b) CTL 2 layers



(c) CTL 3 layers

(d) CTL 4 layers

Figure 3.5: Evolution of the loss during training for a few stock examples of the proposed model with (a) CTL 1 layer, (b) CTL 2 layers, (c) CTL 3 layers and (d) CTL 4 layers.

3.2 DeConDFFuse : Predicting Drug-Drug Interaction using joint Deep Convolutional Transform Learning and Decision Forest fusion framework

Let's move on to the next supervised framework - DeConDFFuse [2], which is again based on CTL. Briefly, the framework jointly trains CTL based network pipelines, fuses them with TL and passes lastly via DF. The technique has been applied to Drug-Drug Interaction prediction. Drug-Drug interactions (DDIs) are the adverse changes or effects, or reactions of one drug due to the recent concurrent use of another drug(s). For example, the drug Ceftriaxone should be avoided in children less than 28 days old if they are receiving or expected to receive IV calcium-containing products. Indeed, it might lead to neonatal deaths resulting from crystalline deposits in the lungs and kidneys, as reported in [114]. Such reaction from DDIs is known as adverse drug reactions (ADRs). ADRs are responsible for the threat to a person's life and inadvertently increase overall healthcare costs.

According to the studies [115, 116], ADRs contribute to more than 20% of clinical trial failures and are considered the highest load in the modern drug discovery process. Serious ADRs can cause severe disability and even death in patients. Also, from study [116], it is observed that approximately 3.6% of all hospital admissions are caused by ADRs in Europe. Up to 10% of patients in European hospitals experience an ADR among those patients. From a financial

[2] P. Gupta, A. Majumdar, E. Chouzenoux and G. Chierchia, "DeConDFFuse : Predicting Drug-Drug Interaction using joint Deep Convolutional Transform Learning and Decision Forest fusion framework". Expert Systems with Applications, Volume 227, 2023, 120238, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2023.120238

perspective, the annual financial cost of drug-related morbidity in the United States (US) was estimated at $\$528.4$ billion in 2016, equivalent to 16% of total US healthcare expenditures that year [117]. It, thus, becomes pertinent to identify, in an exhaustive manner, the DDIs that could cause ADRs. This might not avoid all unanticipated drug interactions, but it can help lower the drug development costs and optimize the drug design process [118].

3.2.1 Literature Review - DDI

DDIs identification is considered a non-trivial problem from the research perspective to be solved in the pharmacology discipline. Many different computational strategies have been investigated in the literature, which are discussed below. Several families of methods can be identified, relying either on statistical machine learning models, graph models, deep learning models and matrix factorization models.

3.2.1.1 Statistical Machine Learning based frameworks

The work [119] proposes similarity-based models that compute similarity scores between drug features like chemical structures, side-effects, targets, pathways, etc. and thereafter performs a probabilistic inference of the DDIs. Researchers have explored Bayesian learning models [120] under statistical learning paradigms. Another work [121] uses a sparse feature learning ensemble method with linear regularization utilizing four drug features - chemical substructures, targets,

enzymes and pathways. In [122], ML algorithms like Naive Bayes, Decision Tree, Random Forest, Logistic Regression, and XGBoost were used with cross-validation with input as SMILE values and interaction features based on CYP450 group.

As also discussed in the previous chapter, all the aforementioned studies utilize statistical machine learning whose performance might depend highly on the quality of features used; thus, it becomes pertinent to explore multiple features than restrict them to some set. Furthermore, overfitting stays a significant issue with these techniques due to their restrictive non-linear mapping and fitting capability.

3.2.1.2 Graph-based frameworks

Graph-based embedding techniques are also gaining momentum in DDIs prediction. With the advent of the availability of biomedical data, researchers are moving toward KGs to populate and complete the available biomedical information. It is done with the help of the large structured databases and texts available publicly [123]. Some works have used the combination of DDI matrix and KG followed by applying ML algorithms [123].

3.2.1.3 Deep Learning based frameworks

Deep Learning is another effective modeling technique that is extensively used in solving most real-world problems these days. It has emerged to be helpful in

the said DDI prediction as well. In [124], the proposed framework integrates the multi-relational and relation-aware network structure representations. Finally, the integrated representations via concatenation are passed through the neural network to get the final DDI predictions. The study [125] proposes attention-based RNNs - LSTM for DDI prediction. In [126], the work utilizes deep Neural Networks based on attention technique for predicting DDIs with features from multiple networks learned using graph embedding techniques.

Further studies are combining KGs and DL to predict DDIs. In the study [127], the DDI matrix and KG form the input to the DL network with CNN and LSTM. KG is input to the network through learned embeddings like ComplEx, TransE, RDF2Vec, etc.

3.2.1.4 Matrix Factorization and multi-modal techniques

Let us also mention recent studies that present matrix factorization as the solution to predict DDIs [128, 129]. Here, the input is the DDI matrix and the similarity scores between the drugs. The pair for which the DDI is to be predicted is treated as a missing value; hence, it is imputed using the inputted similarity scores. Then some works use the Triple Matrix Factorization also [130, 131]. Some researchers have even proposed multi-modal techniques to predict DDIs. The study that used this technique learned the unified drug representations from multiple drug feature networks simultaneously using multi-modal deep auto-encoders. Then, they applied four operators on the learned drug embeddings to represent drug-drug pairs, and finally, they used an RF classifier to train models

for predicting DDIs [132].

Several aforementioned studies are based on different categories like similarity-based, network-based, graph-based, etc., but none guarantees the distinctive Learning of features.

3.2.2 Proposed Formulation

In this section, the proposed work is discussed. A fusion framework is presented that combined the benefits of the recently established multi-channel, unsupervised, fusion-based representation learning framework - DeConFuse [69] and jointly optimized a decision forest with the binary decision that gives the final DDI Predictions. Such a solution has been successfully used before in Deep Neural Decision Forest (DNDF) framework [1]. Also, it is already known that DeConFuse architecture is unsupervised. The aim was to propose a supervised version of this architecture. The recently established supervised version of this framework, namely SuperDeConfuse [57] is just explained in previous few sections. However, the supervision in SuperDeConFuse was incorporated by using cross entropy loss in the optimization objective and softmax at the end of its architecture. Here, the goal was to guide the supervision through a random decision forest. The proposed solution did not utilize features/representations from CNN but instead from the DeConFuse network based on deep CTL through linear transform learning. The latter's advantage was that it promoted distinct filters/transforms, which was not guaranteed with CNNs. Such a benefit helped in learning distinct and interpretable representations. These representations were

further guided by the predictions from the decision forest, whose parameters are jointly optimized. This joint optimization was helpful as the representations learned are guided not only by deep CTL and fusion objectives but also by decision forests. Thus, the representations that would have been fed to the Decision Forest (DF) just like a normal input to it were learned better by its feedback during backpropagation of error to the previous layers' neurons in the joint end-to-end solution that was not available with the piecemeal approach. Further, thus, this double guided (deep CTL based fusion + DF) supervised learned representations helped correctly identify many known-to-interact (1) DDIs, as corroborated by the experimental results discussed in section 3.2.4 later. The details of the DeConFuse network can be referred from Chapter 2. Let's briefly discuss the latter framework - DNDF and, finally, mention the details of the combined fusion framework.

3.2.2.1 DNDF Framework

The DNDF framework from [1] is introduced in this section, which is the last brick of the DDI pipeline. It is different from conventional deep neural networks as it outputs the final predictions from the decision forest, and their split (decision nodes) and leaf (prediction) nodes' parameters are jointly and globally optimized. The technique is stochastic and differentiable, thus giving a backpropagation compatible version of decision trees that guides the representation learning in lower layers of deep CNNs. This reduced the uncertainty on routing decisions of a sample taken at the split nodes, such that the globally defined loss function is

minimized. For the leaf nodes, the optimal predictions are achieved by reducing the convex objective function, which does not require step size selection. Further, the objective function is explained briefly.

Decision Trees with Stochastic Routing

Consider a classification problem with input space χ and finite output space Y. A decision tree comprises decision (or split) and prediction (or leaf) nodes. Decision nodes, let's say, indexed by N are internal nodes of the tree, and prediction nodes are indexed by $\mathcal{L}$, i.e., terminal/leaf nodes of the tree. Each prediction node $\ell \in \mathcal{L}$ is associated with a probability distribution $\pi_\ell = (\pi_{\ell_y})_{y \in Y}$. Each decision node $n \in N$ is assigned a decision function $d_n(\cdot; \theta) : \chi \to [\,0, 1]$ parameterized by θ, which routes the samples along the tree branches. When a sample $x \in \chi$ reaches a decision node n, it will be directed to the left or right subtree based on the output of the function $d_n(x; \theta)$. Here, it is a probabilistic routing where the routing direction is the output of a Bernoulli random variable with mean $d_n(x; \theta)$. As the sample ends in a leaf node ℓ, the related tree prediction is given by the class-label distribution $\pi_\ell = (\pi_{\ell_y})_{y \in Y}$. In the case of stochastic routings, the leaf predictions will be averaged by the probability of reaching the leaf. Thus, the final prediction for a sample x from tree D with decision nodes parameterized by θ is given as:

$$(\forall y \in Y) \qquad \mathbb{P}_D[\, y \mid x, \theta, \pi] \,= \sum_{\ell \in \mathcal{L}} \pi_{\ell_y}\, \mu_\ell(x \mid \theta) \tag{3.6}$$

where $\pi = (\pi_\ell)_{\ell \in \mathcal{L}}$. Here above, $\mu_\ell(x \mid \theta)$ is regarded as the routing function providing the probability that sample x will reach leaf ℓ. Note that $\Sigma_\ell \mu_\ell(x \mid \theta) = 1$ for any $x \in \chi$.

For an explicit form for the routing function, the following binary relations that depend on the tree's structure are given as: $\ell \swarrow n$, which is true if ℓ belongs to the left sub-tree of node n, and $n \searrow \ell$, which is true if ℓ belongs to the right sub-tree of node n. Hence, these relations can be exploited to express μ_ℓ as:

$$\mu_\ell(x \mid \theta) = \prod_{n \in N} d_n(x;\theta)^{\mathbb{1}_{\ell \swarrow n}} \, \bar{d}_n(x;\theta) \tag{3.7}$$

where $\bar{d}_n(x;\theta) = 1 - d_n(x;\theta)$, and $\mathbb{1}_P$ is an indicator function conditioned on the argument P. Although the product in (3.7) runs over all nodes; however, only decision nodes along the path from the root node to the leaf ℓ contribute to μ_ℓ, because for all other nodes $\mathbb{1}_{\ell \swarrow n}$ and $\mathbb{1}_{n \searrow \ell}$ will be both 0 (with the assumption $0^0 = 1$). See Figure 3.6.

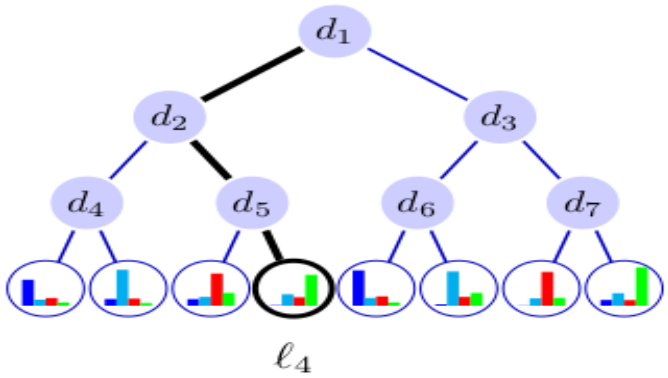


Figure 3.6: Each node $n \in N$ of the tree performs routing decisions via function $d_n(\cdot)$. The black path shows an exemplary routing of a sample x along a tree to reach leaf ℓ_4, which has probability $\mu_{\ell_4} = d_1(x)\bar{d}_2(x)\bar{d}_5(x)$. *Image taken from [1].*

Decision functions deliver a stochastic routing with decision functions defined

as follows:

$$d_n(x;\theta) = \sigma(f_n(x;\theta)), \tag{3.8}$$

where $\sigma(x) = (1 + e^{-x})^{-1}$ is the sigmoid function, and $f_n(\cdot;\theta) : \chi \to \mathbb{R}$ is a real-valued function depending on the sample and the parameterization θ.

A forest is an ensemble of decision trees $\mathcal{F} = \{D_1, \ldots, D_k\}$, which delivers a prediction for a sample x by averaging the output of each tree, i.e.

$$(\forall y \in Y) \qquad \mathbb{P}_{\mathcal{F}}[y \mid x, \theta, \pi] = \frac{1}{k} \sum_{h=1}^{k} \mathbb{P}_{D_h}[y \mid x, \theta, \pi]. \tag{3.9}$$

Note that the tree parameters are omitted for notational convenience.

Learning Trees by Back-Propagation

Learning a decision tree, for which the model is explained in the previous sections, requires estimating the decision node parameterizations θ and the leaf predictions π. The parameters θ are estimated using the Minimum Empirical Risk principle with respect to a given data set $\mathcal{T} \subset \chi \times Y$ under the log-loss (also known as the cross-entropy loss), i.e., minimizers of the following risk term are searched:

$$F_{\text{tree}}(\theta; \pi; \mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{(x,y)\in\mathcal{T}} -\log(\mathbb{P}_D[y \mid x, \theta, \pi]) \tag{3.10}$$

The forest is learned by considering the ensemble of trees $\mathcal{F}$, where all trees can possibly share the parameters in θ. Still, each tree can have a different struc-

ture with a different set of decision functions and independent leaf predictions π. The illustration of the forest of decision trees taking the parameters θ and computing routing decisions and prediction nodes probabilities can be referred to from Figure 3.7. Thus, for the forest, the empirical risk is minimized as:

$$F_{\text{forest}}(\theta; \pi; \mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{(x,y)\in\mathcal{T}} -\log(\mathbb{P}_{\mathcal{F}}[y \mid x, \theta, \pi]). \tag{3.11}$$

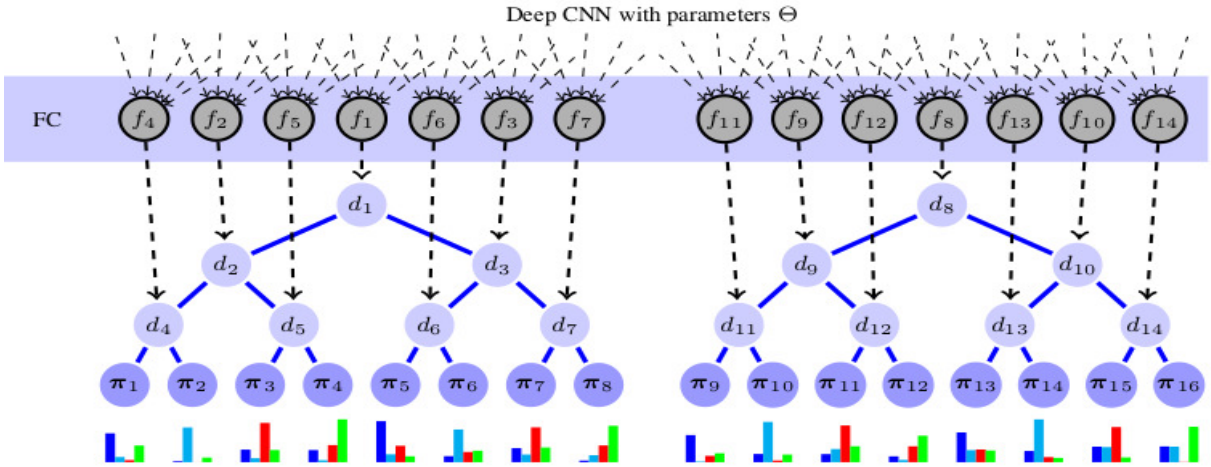


Figure 3.7: illustration of how to implement a deep neural decision forest (DNDF). Top: Deep CNN with a variable number of layers, subsumed via parameters θ. FC block: Fully Connected layer used to provide functions $f_n(\cdot;\theta)$, described in Equ. 3.8. Each output of f_n is brought in correspondence with a split node in a tree, eventually producing the routing (split) decisions $d_n(x) = \sigma(f_n(x))$. The order of the assignments of output units to decision nodes can be arbitrary (the one shown allows a simple visualization). The circles at the bottom correspond to leaf nodes, holding probability distributions π_ℓ. *Image taken from [1].*

A two-step optimization strategy minimizes the above function, with alternate updates of θ and π explained in further sections.

Learning decision nodes

Each function f_n is parameterized by $theta$ on which all decision functions depend as shown in equation 3.8. To minimize the risk term with respect to

$theta$ for a given π by employing Stochastic Gradient Descent (SGD) approach. The update is given as :

$$\begin{aligned}\theta^{(t+1)} &= \theta^{(t)} - \eta \frac{\partial R}{\partial \theta}(\theta^{(t)}, \pi; \mathcal{B}) \\ &= \theta^{(t)} - \frac{\eta}{|\mathcal{B}|} \sum_{(x,y)\in\mathcal{B}} \frac{\partial L}{\partial \theta}(\theta^{(t)}, \pi; , x, y)\end{aligned} \tag{3.12}$$

where $\eta > 0$ is a learning rate and $\mathcal{B} \subseteq \mathcal{T}$ is a random subset or mini-batch. A momentum term was also used to smooth out the gradients' variations not shown explicitly. The gradient of the loss L with respect to θ can be decomposed by the chain rule as follows :

$$\frac{\partial L}{\partial \theta}(\theta, \pi; , x, y) = \sum_{n\in\mathcal{N}} \frac{\partial L(\theta, \pi; , x, y)}{\partial f_n(x; \theta)} \frac{\partial f_n(x; \theta)}{\partial \theta} \tag{3.13}$$

The gradient term that depended on decision forest was given by

$$\frac{\partial L}{\partial f_n(x; \theta}(\theta, \pi; , x, y) = d_n(x; \theta) A_{n_r} - \bar{d}_n(x; \theta) A_{n_l}, \tag{3.14}$$

Here n_l and n_r represent left and right children of node n respectively, and A_m for a node $m \in N$ is defined as :

$$A_m = \frac{\sum_{l\in\mathcal{L}_m} \pi_{l_y} \mu_l(x|\theta)}{\mathbb{P}_T[y|x, \theta, \pi]} \tag{3.15}$$

with $\mathcal{L}_m \in \mathcal{L}$ denoted the set of leaves held by the sub-tree rooted in node m.

Learning prediction nodes

After the understanding of learning θ from previous section, let's learn about prediction nodes. The risk term in 3.10 with respect to π when θ is fixed was learnt which is a convex optimization problem with a global solution. Here, all the leaf nodes were estimated jointly. The iterations for the updates read as :

$$\pi_{l_y}^{(t+1)} = \frac{1}{Z_l^{(t)}} \sum_{(x,y') \in \mathcal{T}} \frac{\mathbb{1}_{y=y'} \pi_{l_y}^{(t)}(x|\theta)}{\mathbb{P}_T[y|x, \theta, \pi^(t)]} \tag{3.16}$$

for all $l \in \mathcal{L}$ and $y \in Y$, where $Z_l^{(t)}$ was a normalizing factor so that $\sum_y \pi_y^{(t+1)} = 1$. Initial $\pi^(0)$ is random, typically $\pi_{l_y}^0 = |Y|^{-1}$. Here, the update of π was interleaved with a whole epoch of stochastic updates of θ.

3.2.2.2 Combined Proposed Framework DeConDFFuse - DeConFuse and Decision Forest

The proposed framework combined the frameworks DeConFuse explained in the previous chapter 2 and DNDF in section 3.2.2.1, respectively. Specifically, instead of utilizing the features from a CNN network, it was proposed to inherit the representations learned from the DeConFuse network to peruse them in the decision forest, i.e., the decision forest was jointly trained and optimized within the DeConFuse network. The DeConFuse network learned the channel-wise representations corresponding to each drug in a drug pair, that is $X^{(c)}$ with $c \in \{1, 2\}$, and finally learned common representation Z from $X^{(c)}$ where the fusion happened. Here, there was no positivity constraint on Z and only on

channel-wise representations $X^{(c)}$.

The representation Z was passed to the DF, where it applied the features mask, i.e., randomly selected the features from the representation that would participate in the decision tree's routing process which fed those selected features to the linear fully connected layer parameterized by θ, i.e., given by the function $f_n(x;\theta_n) = \theta_n^\top x$. The number of features involved was provided by the feature ratio. After that, the sigmoid activation was applied as given in Eq. 3.8. Then the routing function was computed, and the prediction probabilities were calculated. Thus, the prediction probabilities having a probability for each class for each tree were likewise obtained. Finally, the probabilities from all the trees of the Forest $\mathcal{F}$ were averaged to get the outcome probability for each of the classes 0 and 1 in this case. The negative log-likelihood loss was computed and back-propagated, which guided the representation learning of the DeConFuse framework and Learning of the parameters θ. The objective function for this framework that combined the idea of DeConFuse and DF can be deduced from 2.10 and 3.11:

$$\underset{T,X,\widetilde{T},Z,\theta,\pi}{\text{minimize}} \underbrace{F_{\text{fusion}}(\widetilde{T}, Z, X) + \sum_{c=1}^{C} F_{\text{conv}}(T_1^{(c)}, \ldots, T_L^{(c)}, X^{(c)} \mid S^{(c)}) + F_{\text{forest}}(\theta;\pi;\mathcal{T}_Z)}_{J(T,X,\widetilde{T},Z,\theta,\pi)}. \tag{3.17}$$

Hereabove, the dataset $\mathcal{T}_Z$ was built with the learned features Z and the known labels. Note that there was no positivity constraint anymore on the learned representations Z.

3.2.3 Experimental Setup

The experiments were conducted on the drug-drug interaction dataset comprising the DDI matrix and bioactivity descriptors/feature vectors for each drug as explained in section 1.3.3. The DDI matrix dataset was divided into training and testing datasets. All the drugs are kept in the training data so that there are 95 samples per drug. Further, out of 95 samples, there are 60% of 1 interactions for each drug (not exceeding half of the 95, i.e., min (60% of 1 interactions, $95//2$)), and the remaining are the samples from 0 interactions. The remaining samples of 0 and 1 interactions per drug are kept in testing data. All the training and test data samples from each interaction category per drug are selected randomly. Also, only one pair of interactions are kept from either the upper or lower triangle of the DDI matrix. Thus, each training and testing sample is the drug pair and its corresponding interaction value, which is called as a label. Approximately there are 1Lac training samples and 4Lac testing samples.

A drug pair was passed as a input to a sample during training. For each drug in a pair, 1D feature vectors, i.e., bioactivity descriptors, were fed to the individual channel/network based on deep CTL, where $L = 2$ represents the number of CTL layers. Thus, the input S gathered the bioactivity descriptors/1D feature vectors of size 384 for each channel corresponding to each drug. Since there were 1D feature vectors for each drug in the drug pair, thus, 1D convolutions were used in each deep CTL network. The two networks' learned features/representations $X^{(c)}$ were flattened and concatenated. Then these features were passed to the linear

Transform learning layer that acted as a fully connected layer where transform $\widetilde{T}$ and common representation Z were learned. Further, the learned representation Z were selectively sent by applying the feature mask to the decision forest. The final predictions were outputted by averaging the predictions from each tree in the decision forest.

The complete architecture is shown in Figure 3.8 and all the architectural and hyperparameter details are given in Table 3.13. In the set of hyperparameters, the atom ratio signified the number of features to be kept in the representation Z; and the feature ratio meant the randomly selected number of features from the representation Z that would participate in the routing decision function of each tree parameterized by θ.

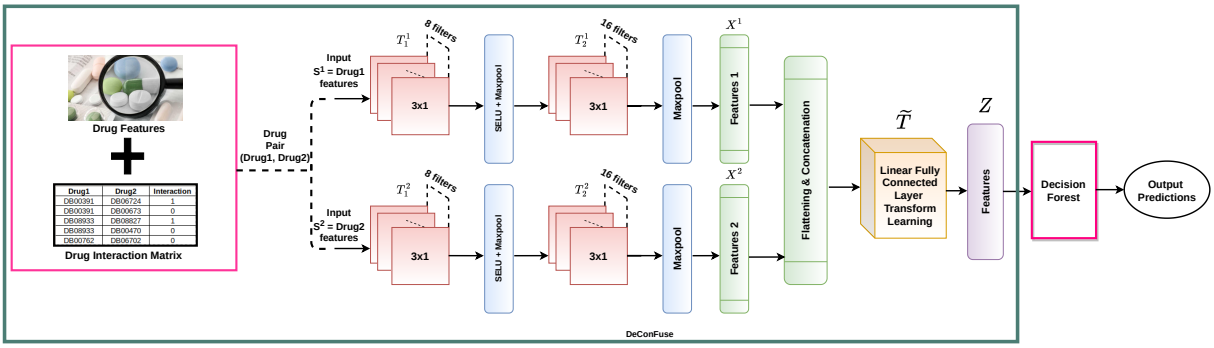


Figure 3.8: DDI prediction using combined DeConFuse and decision forest architecture- DeConDFFuse. Here $C = 2$, the number of networks/channels via each of which a drug in the drug pair is passed along with its bioactivity descriptors/ features vector, respectively.

The results were compared with three state-of-the-arts/benchmark techniques, namely:

- **KGNN**: This technique was used to build the Knowledge Graph (KG) and passes the DDI matrix and KG to the Graph Neural Network (GNN). It focused on neighborhood sampling and aggregates entities and their

Table 3.13: DDI Prediction DeConDFFuse Architecture Details

Parameter	Value
Layer Wise Hyperparameters	
Layer1 - Convolution (CTL)	(1,16,3,1,1)[1]
Maxpool	(2,2)[2]
Layer2 - Convolution (CTL)	(1,32,3,1,1)[1]
Maxpool	(2,2)[2]
atom ratio	0.75
Decision Forest Hyperparameters	
#**Trees**	90
tree depth	10
feature ratio	0.5
Other Model Hyperparameters	
Epochs	75
Learning Rate	0.01
μ	1e-05
λ	0.0001
batch size	4096
weight decay	1e-05
Optimizer Hyperparameters	
Optimizer Used	Adam
Ams grad	True
Learning rate	0.01
betas	(0.9, 0.999)
eps	1e-08

[1] (in_planes, out_planes, kernel_size, stride, padding)
[2] (kernel_size, stride)

neighborhood representation into a single vector in 3 ways - sum, concat, and neighbor [133].

- **Conv-LSTM**: This technique used the DDI matrix and KG as the input to the DL network with a CNN and LSTM. KG was input to the network in the form of learned embeddings like ComplEx, TransE, RDF2Vec, etc. [127]. It is compared against the embedding that gave the best results in this work, i.e., ComplEx embedding.

- **Graph Embedding DDI**: This technique used KG and DDI matrix as input

but experiments with many different types of embedding techniques. Then each of these embeddings, one by one, was passed to machine learning techniques like Random Decision Forest (RF), Gaussian Naive Bayes (GNB), and Logistic Regression (LR) [123]. Here also, the embedding type Skip Gram was used, which gave the best results in their study.

For all three benchmarks - KGNN, KG Conv-LSTM, and Graph Embedding DDI- the same DrugBank IDs were used as present in the considered training and testing samples. Since these methods relied on KGs, the bioactivity descriptors/features were not used but recreated KG and embeddings for the dataset used to run these benchmarks.

3.2.4 Results and Analysis

The prediction results were evaluated using the classification metrics - AUPRC, AUC_ROC, F1 Score, Precision, Recall, and Accuracy. Except AUPRC, all other metrics have been explained previously in the chapter 2. Thus, here AUPRC is only explained as:

AUPRC : it is the area under the graph constructed by calculating and plotting the precision against the recall for a single classifier at a variety of thresholds. The higher the AUPRC score, the better a classifier performs for the given task. It summarizes a precision-recall curve as the weighted mean of precision achieved at each threshold, with the increase in recall from the previous threshold used as the weight. Thus, it is a kind of weighted-average precision across all thresholds.

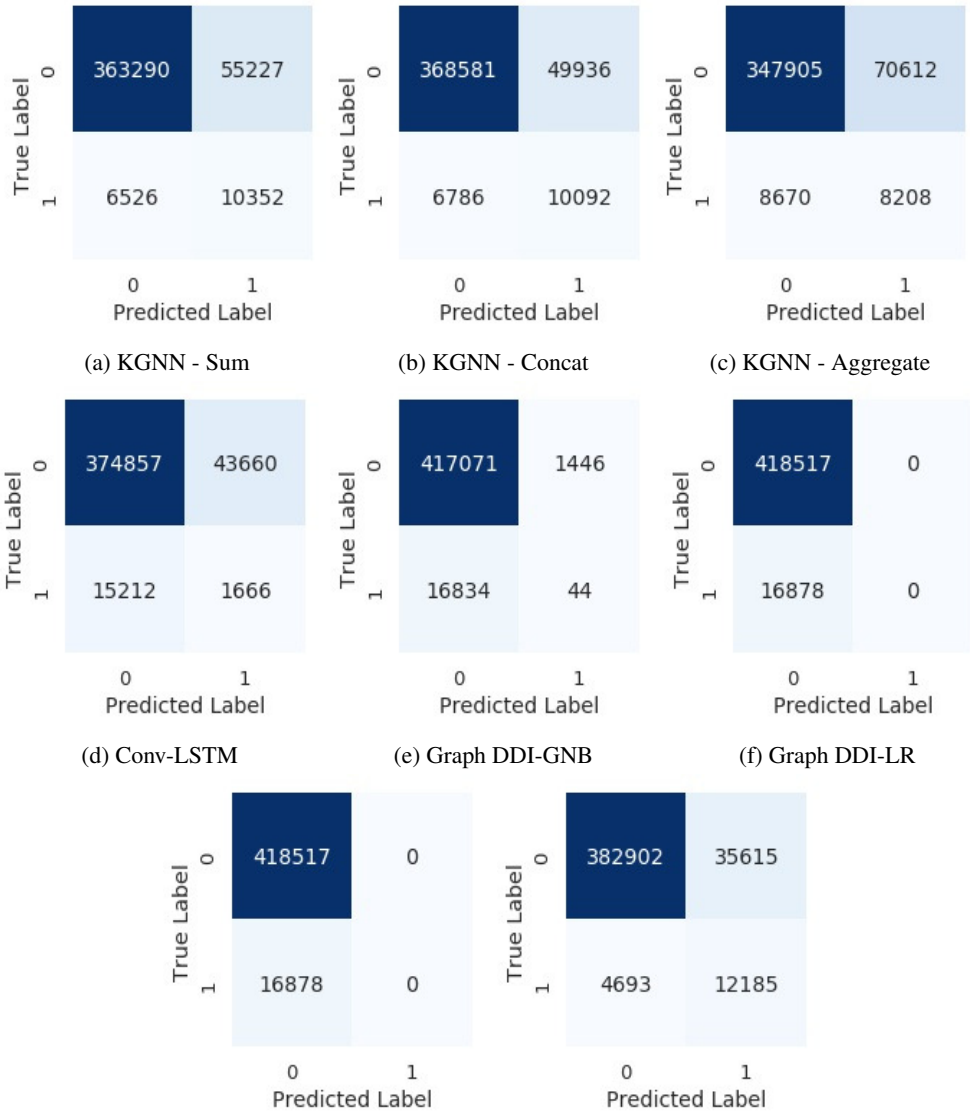
All the metrics were computed except Accuracy as weighted metrics since there was a huge class imbalance between 0 and 1 labels. The following Table 3.14 contains the values of the said evaluation metrics:

Table 3.14: **DDI Prediction Results**

Method	Sub Method	Accuracy	F1	Precision	Recall	AUC ROC	AUPRC
KGNN	**Sum**	85.8168	89.5672	95.0392	85.8168	82.6508	18.6945
	Concat	86.9723	90.2730	95.0375	86.9723	83.5235	19.8427
	Neighbor	81.7908	86.9563	94.1900	81.7908	74.379	10.7655
Conv-LSTM ComplEx	-	86.4785	89.3325	92.5174	86.4785	49.597	3.8164
Graph DDI Skip Gram	**GNB**	95.8015	94.0807	92.5087	95.8015	50.1899	3.8546
	LR	**96.1235**	**94.2236**	92.3973	**96.1235**	50.0603	3.8583
	RF	**96.1235**	**94.2236**	92.3973	**96.1235**	50.0934	3.8439
DeConDFFuse (Ours)	-	90.7422	92.7777	**95.9478**	90.7422	**91.4453**	**34.0847**

Also, confusion matrices for each method were computed. They are displayed in Figure 3.9.

From Table 3.14, it is seen that benchmark Graph DDI gave the best values in terms of Accuracy, F1 Score and Recall, and the proposed method for Precision, AUC ROC, and AUPRC. However, no single Benchmark worked well in terms of all the classification metrics used for evaluation. In fact, the next best performance in terms of Accuracy and F1 was given by the proposed method. Despite the highest F1, Accuracy, and recall values, Graph DDI failed to achieve the highest values for AUC -ROC and AUPRC, which are considered



(a) KGNN - Sum (b) KGNN - Concat (c) KGNN - Aggregate

(d) Conv-LSTM (e) Graph DDI-GNB (f) Graph DDI-LR

(g) Graph DDI-RF (h) Proposed method - DeConDF-Fuse

Figure 3.9: Confusion matrices for different benchmarks and the proposed method- DeConDFFuse

more relevant and important metrics for the performance evaluation in the case of binary classification. The reason for the same can be observed with the help of the confusion matrices in Figure 3.9.

Here, it can be seen that with the proposed method, the highest number of known-to-interact interactions (1) have been predicted correctly than any other benchmarks. Also, except for Graph DDI, the False positives, i.e., classifying

0 as 1, were lesser with the proposed method than the other two benchmarks. Here, the former task of classifying the known-to-interact drug interactions is more important to prevent ADRs, as explained before and which Graph DDI did not achieve at all or was nearly negligible. Thus, with the proposed method, the former task of identifying known-to-interact interactions was accomplished better than any other benchmark. For the false positives, too, it gave good performance compared to the other benchmarks except for graph DDI. The latter is the reason why Graph DDI has three metric values higher than the proposed method. With the proposed approach, though the number of False positives was higher than Graph DDI, however, it was not necessary that these False positives were completely incorrect. The reason is that the 0 interactions did not signify no interaction between those two drugs. It meant either known-not-to-interact or unknown.

In summary, Graph DDI classified almost all 0 (known-not-to-interact or unknown) interactions correctly; still, it did not correctly classify 1 (known-to-interact) interactions that were against the study's objective, i.e., to identify the known-to-interact DDIs to avoid ADRs. With the proposed method, both types of interactions were classified reasonably well. It was also the stable method corroborated by the classification metrics, as it gave good performance in all the metrics. Hence, the proposed framework performed superior to the benchmarks.

Additionally, the comparison was done among representations/features learned from benchmarks and the proposed framework - DeConDFFuse (DCDF). The proposed method performed better due to the kind of representations/features

learned from the CTL based network. Each of the benchmarks was carefully examined. In KGNN, Knowledge Graph features, different aggregation techniques and Graph Neural Network are utilized. The latter had a lot of parameters and computation cost since the neural network connects neurons in each layer with every neuron in the preceding and consecutive layers. Also, it had no distinctiveness guarantee for the kind of weights learned. The poorest performance from Graph DDI was due to a lack of learning ability of the traditional machine learning algorithms utilized in this framework after learning the embeddings from KG. Lastly, with the Conv-LSTM framework, all the disadvantages of CNN discussed in section 3.2.1.3 are applicable. Thus, the representations learned with it might have redundancy leading to inferior performance. Therefore, from overall performance, it could be concluded that the representations learned from the proposed method were better than the benchmarks.

The optimizer used for updating all the parameters of the framework except probability distribution π of the decision forest is Adam, which uses the automatic differentiation in PyTorch for gradient computation. The hyperparameters like learning rate, betas, and eps, etc. associated with it are mentioned in Table 3.13. Loss plots were also plotted with the proposed technique, which can be referred to from Figure 3.10. It can be seen that with Adam optimizer, the proposed solution converged to the point of stability.

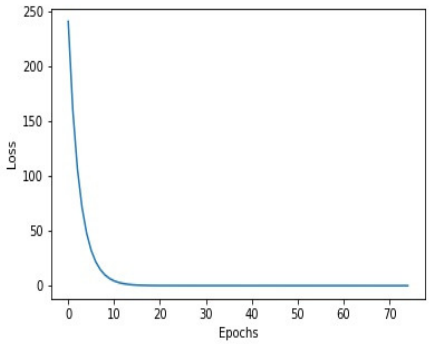


Figure 3.10: Loss plot with the proposed method - DeConDFFuse.

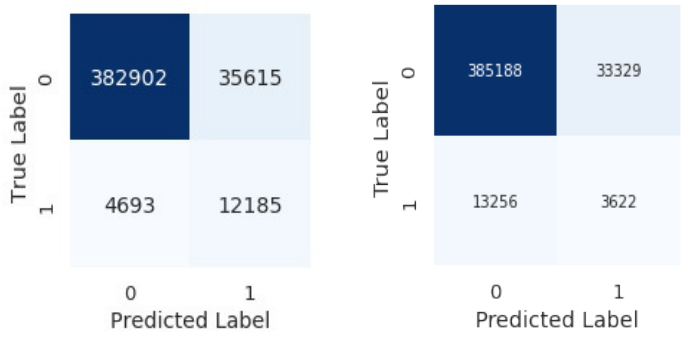
3.2.4.1 Ablation study

The experiments were further carried out to understand the architecture in detail. It involved the performance comparison between the proposed joint model - DeConDFFuse and the piecemeal version. The latter trained to learn the representations Z in an unsupervised manner versus a supervised way. Next, these learned representations Z are fed to the separate module Random Decision Forest where Z are treated as regular input to the system along with the labels. The hyperparameters for DCTL networks are the same as the proposed solution. However, for the RDF part, the hyperparameters were best tuned and were set to values: $\#trees = 5$, $tree\_depth = 70$ and $random\,state = 11$. The two systems' performance was evaluated using the same metrics and are reported in Table 3.15 below.

Table 3.15: **Comparative Results with DeConDFFuse and Piecemeal approaches**

Method	Sub Method	Accuracy	F1	Precision	Recall	AUC ROC	AUPRC
DeConDFFuse (Ours)	-	**90.7422**	**92.7777**	**95.9478**	**90.7422**	**91.4453**	**34.0847**
piecemeal	-	89.2075	91.1640	93.3055	89.3005	62.9022	6.9772

From the above table, it can be clearly seen that the performance of the proposed solution is better than the piecemeal version. Further, it can be better visualized with the help of the confusion matrices given below in Fig. 3.11.



(a) Proposed method - DeConDF-Fuse

(b) Piecemeal

Figure 3.11: Confusion matrices for the proposed method - DeConDFFuse and Piecemeal approach

On comparing the two versions, i.e., the proposed framework with the piecemeal approach, it can be clearly seen that the piecemeal version is not as good as the proposed method in predicting known-to-interact interactions labeled by 1. Although the performance concerning predicting interactions labeled by 0 is slightly better than the piecemeal version; however, the critical point is to predict the known-to-interact interactions (1) that are responsible for ADRs. The latter is poorly predicted with the piecemeal approach compared to the proposed approach. Computation metric wise too, the results are better with the

proposed model. It can be concluded that the joint optimization and training of Decision Forest (DF) with the DCTL-based fusion networks in the proposed solution results in better representations due to added guidance from the DF that is missing in the piecemeal version. This added guidance is the backpropagation of the error from DF classifier to the previous neurons which is missing in the piecemeal approach where the input, i.e., representations Z are treated as normal input without feedback to the system.

3.3 Discussion

So, two supervised frameworks - SuperDeConFuse (SDCF) and DeConDFFuse (DCDF) are discussed- both based on CTL and extended DeConFuse in this chapter. First, the SuperDeConFuse network was discussed, a deep fusion end-to-end framework for processing stock trading data, leading to very good performance. In particular, the classification results are better with the proposed SDCF model than with the 1-D CNN approach. Also, the features X_c visualized for each channel and each method indicated better feature learning with SDCF. The results have shown that the presented solution (SDCF) is superior to CNN and other state-of-the- art techniques in this problem. Additionally, the framework handled the dead neuron problem via positivity constraint.

Currently, the shortcoming of the model is that it takes slightly more time than CNN for its training. Thus, techniques will be investigated that reduce the time complexity of the proposed framework to make it more efficient from this

viewpoint in the future.

The purpose of this framework was to show by means of several experiments that it is an effective tool for predicting stocks. However, stock price prediction might be seen as a too rudimentary problem in financial analytics. As a next step, it will be to investigate the use of the proposed algorithm to study if it can emulate (human) expert-like suggestions. For example, fund managers suggest 'buy stock XYZ at a price ABC' or 'sell stock ZYX at a price CBA.' It will be interesting to see if this algorithm can make such predictions given a time horizon in the future. If possible, in the future, the present algorithm will be extended to emulate more abstract financial operations such as 'hedging (longs and shorts)'.

Secondly, DeConDFFuse was introduced, which is based on the proposed DeConFuse framework and combined it with Decision Forest previously established in the DNDF framework. This framework is a deep supervised fusion end-to-end framework for processing 1D multi-channel drug data. Unlike other deep learning models that separately use conventional machine learning algorithms like RDF, the proposed framework is jointly optimized and is not piecemeal. It has been applied for the binary classification task of DDI prediction leading to good performance. The advantages of the proposed framework are the benefits of CTL based networks. That is, it helped learn non-redundant common representation for the problem where there are two drugs in a drug pair that is also guided by the jointly optimized Decision Forest loss. Reasonably well performance is achieved compared to the state-of-the-art(s).

In the future, there is a scope to improve performance by reducing the number of false positives. Also, the current solution to the DDI problem considers the event when two drugs are administered together. However, a combination of more than two drugs is routinely used. Thus, another extension will be the capability of the proposed framework to handle more than two drugs combinations in the future. This can be done with the proposed architecture by increasing the number of channels per increase in the number of drugs. Lastly, although the proposed architecture has been applied to drug-drug interaction, it is flexible enough to be applied to other biomedical interaction problems. In the future, other areas can be explored such as drug-target prediction [134–136], protein-protein interaction [137–139] and drug repositioning [140].

In summary, both the frameworks - SuperDeConFuse and DeConDFFuse presented are the supervised versions of CTL jointly trained and optimized by cross-entropy loss and decision forest, respectively. Both performed well in their respective problems by leveraging CTL's benefits. However, each of the proposed frameworks offers scope for improvement as discussed above for each separately.

Chapter 4

Multiview Clustering Framework based on CTL - DeConFCluster

With the rapid increase in data collection sources and volume, the exploration of multiview data has become popular. Multiview data is referred to as the data collected from the same data source but with different angles or different perspectives. For example, the same news is advertised or published in different media with different content; the same statement is labeled with different tags by different individuals, and the same image is captured using different features. Multiview data is richer and more informative but more complex than single-view data. In multiview data, the data belonging to each view has information related to different contexts and also has some complementary information. Clustering is a category of unsupervised learning approach in which the data instances are grouped into several groups or clusters based on the various features of the data instances. Thus, multiview data clustering requires exploring and integrating multiple views of the data to perform the grouping of data instances in possible

clusters.

Multiview data knowledge extraction is vital in big data mining and analytics nowadays. In this regard, many recent works suggest CNN based clustering objectives. These generally lie on the encoder-decoder framework. In such a work, the clustering loss is included after the encoder network, which ensues the problem of additional training of a decoder network and hence, incurs extra learning of weights. In data-constrained scenarios, this can make the model prone to overfitting. Also, some works learn representations independently and apply clustering algorithms like K-Means in a piecemeal fashion which may lead to representations being less effective for clustering task.

The success of CTL based unsupervised and supervised frameworks for performing classification and regression tasks have been already witnessed in last few chapters. In this chapter, an unsupervised multi-channel multiview clustering framework based on Deep Convolutional Transform Learning (DCTL) - DeConFCluster is introduced that bridges all the gaps mentioned earlier, namely

1. it avoids additional decoder training,
2. it learns distinct transforms and
3. it learns representations from joint training of deep CTL multiview layers and K-Means algorithm.

The proposed framework is evaluated on four standard multiview clustering datasets. The results demonstrate that the proposed framework outperforms the

state-of-the-art multiview deep clustering approaches. This chapter is further organized into sections, with the first section 4.1 discussing the current related works for MVC. Next, the proposed formulation is explained in section 4.2. The experimental evaluations and results are discussed in sections 4.3 and 4.4 respectively, followed by a brief discussion in section 4.5.

4.1 Literature Review

Multiview clustering clusters subjects into subgroups using multiview data and has gained significant attention rapidly as it caters to solving real-world problems that fall under big data analytics. Recently many solutions have been proposed to perform the same. These are broadly classified into two categories generative and discriminative approaches. The generative approaches try to learn the underlying data distribution. These use generative models with each model representing the individual view and then find the clustering solution. In contrast, discriminative approaches seek to optimize an objective function with pairwise similarities. The average similarity in intra-clusters and inter-clusters is minimized and maximized respectively [141]. The former usually includes expectation maximization and mixture models. The latter, being larger in number, can be further categorized into sub-categories like multiview spectral clustering, multiview subspace clustering, multiview non-negative matrix factorization clustering, multi-kernel clustering, Canonical Correlation Analysis (CCA), etc. [141].

In generative approaches, the work in [142] assumes independent views and

adopts multinomial distribution for the document clustering problem. Similarly, based on different assumptions and criteria, two versions of the multiview EM algorithm for finite mixture models are proposed in [143]. Using Convex Mixture Models (CMMs) for single-view clustering, the multiview version proposed in [144] could find the global optimum. It also avoided the initialization and local optima problems of standard mixture models, as the latter requires multiple EM algorithms executions. The major issue with EM based algorithms is their slow convergence, and convergence to local optima. The second issue in EM based algorithms is that in some scenarios, the E-step and M-step could be unmanageable analytically since it requires both forward and backward probabilities versus the numerical optimization that requires only forward probability.

Next, let's discuss the multiview spectral clustering method in discriminative approaches. This method obtains a common clustering result and assumes that the same or similar eigenvector matrix is shared among all views. There are two characteristic methods. First is co-training spectral clustering [145–148] when both labeled and unlabelled data are available. Second is co-regularized spectral clustering [149, 150], which is a semi-supervised learning technique. The objective function generally requires the difference between the predictor functions of the two views to be minimized.

There are methodologies based on subspace clustering in the multiview data [151–153]. It requires finding the underlying low dimensional common subspace from each view which is, in general, obtained by making each of the view's coefficient matrix as similar as possible. The other works suggest Non-Negative

Matrix Factorization (NMF) that seeks two non-negative matrix factors called basis and indicator. In the case of MVC, some studies point to learning a common indicator matrix across each view [154, 155] for NMF. Some works propose using multiview K-Means clustering to deal with the extensive data. These works use K-Means as it is computationally less expensive than eigen-decomposition. In [156], authors proposed a multiview K-Means clustering method that adopted a common indicator matrix across different views. Besides Non-negative Matrix Factorization (NMF), the authors in [157] introduced a categorical utility function to measure the similarity between the indicator matrix from each view and the common indicator matrix and proposed a consensus based MVC method.

Also, there are methods in which direct view combination via a kernel is used as a common approach to perform MVC. Usually, it is done by designating a kernel for each view and then combining these kernels in a convex combination [158–160]. Another technique - CCA combines multiple views after projection [161, 162]. All methods mentioned earlier have achieved satisfactory performance for the clustering task. But, it may be challenging to handle the data with high-dimensional features and nonlinear property using the above stated methods since they majorly adopt shallow and linear embedding functions to reveal the intrinsic structure of the multiview data.

Recently, graph based MVC has also gained momentum. The authors in [59] proposed a solution wherein the graph matrices of multiple views are combined into a unified graph matrix by generating the Similarity Induced Graph

(SIG) matrices for all the available views. Then the rank constraint on the graph Laplacian matrix is applied, and the number of connected components are produced from the unified graph, which gives the final number of clusters.

Deep learning has emerged as a highly utilized technique to solve almost all real-world problems and is used in the case of MVC. In [163], multiple autoencoders are utilized for multiview data to generate multiple latent representations and apply heterogeneous graph learning to fuse the generated latent representations followed by the K-Means network for the final clusters. Further, in the study [164], based on autoencoders, Deep Embedded Multiview Clustering with collaborative training (DEMVC) is proposed. It utilizes complementary and consensus information from multiple views and collaboratively learns the deep latent feature representations and clustering assignments.

A Graph Neural Network (GNN) [165] is applied to deep representation-based MVC to completely benefit from the features embedded in the attributed multiview graph data. Further, the work in [166] used Graph Convolutional Network (GCN) as an encoder with the most reliable view as input. In another study, multiple GCN decoders capture the view-consistent low-dimensional feature representation among different views [167]. Here, the issue is with the additional weights training incurred from the decoder network, which could lead to overfitting in data-constrained scenarios [168]. Also, another shortcoming of existing solutions is due to CNN. Additionally, CNN ends up in a trivial solution without an output. Employing Deconvolutional layers is the lone way to prevent the trivial solution. However, even using the mentioned solution, there

are chances of over-fitting. Further, CNNs do not guarantee the distinct learning of filters that may lead to redundant representations/features which may reduce the performance of the task at hand.

The work in [169] embedded K-Means clustering in the Transform Learning framework and trained in a joint end-to-end fashion. Also, DCTL was utilized to perform clustering by jointly training it with K-Means to perform single-view clustering [168]. On the contrary, in this chapter, an MVC framework is introduced that jointly trains and optimizes DeConFuse and K-Means clustering modules in this work. It is named as DeConFCluster and it overcomes the aforementioned shortcomings.

4.2 Proposed Formulation - DeConFCluster: Deep Convolutional Transform Learning based Multiview Clustering Fusion Framework

In this section, the proposed work is discussed. The proposed framework is an unsupervised multi-channel fusion framework called DeConFCluster to perform MVC. It extends the previously established works - Deep CTL based K-Means clustering framework [168] utilized for single-view clustering and DeConFuse framework for MVC. The latter framework has already been discussed in Chapter 2. Next, there is a brief discussion of the other prior method mentioned and finally, the proposed formulation is explained in subsequent sections.

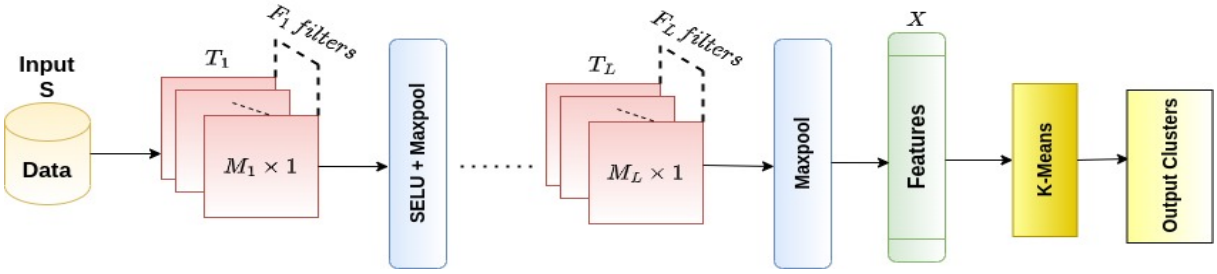


Figure 4.1: DCKM architecture. L represents number of DCTL layers, M_l^c - filter size and F_l^c - #filters of the respective layer l and channel c.

4.2.1 Deep CTL Based K-Means Clustering Framework

This framework extended the Deep CTL (DCTL) approach by adding the K-Means loss at the end. The DCTL approach from Section 2.2.2 explained in Chapter 2 was extended by jointly training and optimizing it with K-Means to perform single-view clustering [168]. The loss formulation after embedding with the K-Means clustering loss [170], was:

$$\underset{T_1,\ldots,T_L,X,H}{\text{minimize}} \; F_{\text{conv}}(T_1,\ldots,T_L,X \mid S) + \beta \underbrace{\|X - XH^\top (HH^\top)^{-1} H\|_F^2}_{\text{K-Means loss}}. \quad (4.1)$$

Here, X is the representation learned, S is the input, $\beta > 0$ is the regularization weight associated with the K-Means clustering loss, and H is the matrix of binary indicator variables such that an entry $h_{ij} = 1$ if x_j belongs to cluster i and 0 otherwise. The architecture is summarized in Fig. 4.1.

4.2.2 Proposed Formulation

Here, the proposed unsupervised fusion framework - DeConFCluster[1] is discussed. Previously, the framework DCKM [168] combined DCTL [67] with K-Means for Single View Clustering (SVC). In contrast, a multiview clustering task is targeted here. Hence, DeConFCluster was a multi-channel clustering framework that extended DeConFuse Network [69] based on DCTL by embedding the K-Means clustering loss as was done in DCKM [168]. Here, fusion was happening that was not present in DCKM. It jointly trained and globally optimized DeConFuse Network and K-Means module. There were as many channels as the number of views in any of the considered datasets, i.e., $C = V$. Each channel was processed based on the DCTL network. This amounted to learning distinct transforms $(T_c)_{1\leq c\leq C}$ and thus, distinct and interpretable representation $(X_c)_{1\leq c\leq C}$, for each channel input $(S_c)_{1\leq c\leq C}$. These channel wise representations were further fused using TL [90] to learn a common representation Z and transform $\widetilde{T}$. This completed the first module of the architecture. The representations are then fed as input to the second part of the framework K-Means clustering module that gives the clustering results. Thus, the representations learned are also guided by the K-Means loss. The learning problem reads:

$$\underset{T,X,\widetilde{T},Z,H}{\text{minimize}} \; F_{\text{fusion}}(\widetilde{T}, Z, X) + \sum_{c=1}^{C} F_{\text{conv}}(T_1^{(c)}, \ldots, T_L^{(c)}, X^{(c)} \mid S^{(c)}) + \beta \|Z - ZH^\top (HH^\top)^{-1} H\|_F^2 \tag{4.2}$$

[1]P.Gupta, A. Goel, A. Majumdar, E. Chouzenoux and G. Chierchia, “DeConFCluster: Deep Convolutional Transform Learning based Multiview Clustering Fusion Framework". 2023. Submitted in IEEE TNNLS

The complete architecture of the DeConFCluster is summarized in the Fig. 4.2

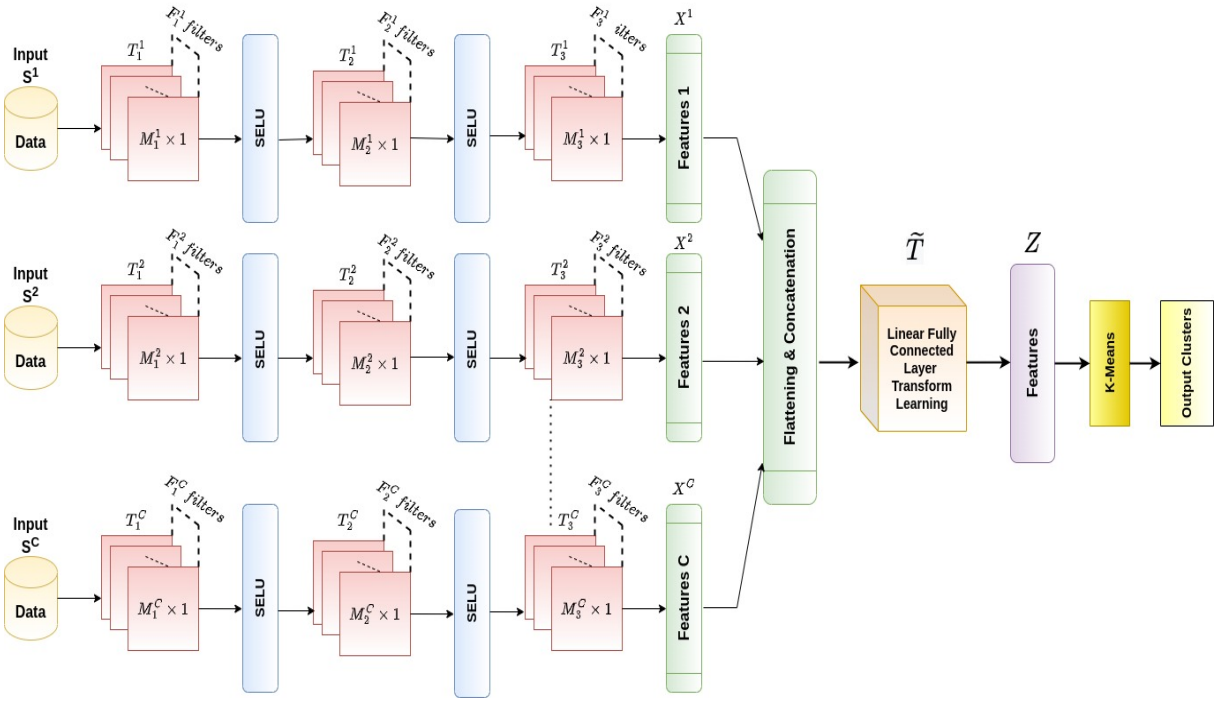


Figure 4.2: Overview of the proposed DeConFCluster architecture. C represents the number of DeepCTL network-s/channels, L is the number of DCTL layers, M_ℓ^c is the filter size and F_ℓ^c is the number of filters of the respective layer ℓ and channel c.

All the variables were learned in an end-to-end fashion. Typically, SGD could be used as an optimizer for all the variables except H. This latter variable was updated directly via K-Means clustering [170] at each iteration using the current Z estimate as an input.

4.3 Experimental Setup

In this section, the performance of the proposed approach is illustrated on various multiview clustering datasets - 100leaves, ALOI, Mfeat and WebKB which have been already discussed in the section 1.3.4. Next, let us explain the network architecture. The network's pipeline consisted of multiple channels wherein

each channel was designated for one of the views of the multiview dataset. Then the representations were learned from these channels' networks that gave the individual view's contribution. Further, these representations were flattened and concatenated to pass through a fully connected layer learned via TL. Here, the common representation was learned across all channels' representations that provided the cross-channel information or shared information from each view. Finally, clusters were obtained by inputting the representation into the K-Means module. The pipeline is shown in Fig. 4.2. The Stochastic Gradient Descent (SGD) algorithm was used as the optimizer and $\lambda = 0.01$, $\mu = 0.0001$ and weight decay as 0.001 for all datasets. There was also another hyperparameter - feature_-ratio that indicated the percentage of features kept in the final representation Z. All other hyperparameters' values are grid-searched and the ones that gave best results are set as the final values. These values can be referred from Table 4.1.

Table 4.1: DeConFCluster hyperparameters for MVC Datasets

Parameter	**100leaves**	**WebKB**	**Mfeat**	**ALOI**
Batch size	1600	203	128	11025
Epochs	25	25	40	25
Learning Rate	5e-6	1e-4	1e-4	5e-6
Kernel Sizes[1]	(3,3,3)	(3,3,3)	(5,3,3)	(3,3,3)
#Filters[2]	(4,8,16)	(4,8,16)	(2,4,8)	(4,8,16)
feature_ratio	0.15	0.15	0.25	0.25
β[3]	1.0	0.5	0.8	0.5

[1] Kernel sizes for DCTL layers 1,2,3
[2] #Filters for DCTL layers 1,2,3
[3] K-Means loss regularizer

The results were compared with four state-of-art works. These are briefly described here that are as follow:

- MCGL: it was a graph based learning method. Starting graphs were learned

using different views' data points that were further optimized with a rank constraint on the Laplacian matrix. Next, optimized graphs were integrated into a global graph. The graph was learned with the same rank constraint on its Laplacian matrix. Cluster indicators were obtained from the global graph only without conducting any graph cut technique and the K-Means clustering [171].

- GMC: in this approach, each view was weighted and the SIG matrices and the unified graph matrix were jointly learned [59]. The latter was obtained by the fusion of the graph matrices of each view.

- DEMVC: this method proposed a framework based on autoencoders. It utilized complementary and consensus information from multiple views and learned the deep latent feature representations and clustering assignments in a collaborative manner [164].

- RRA-MVC: this technique proposed a simple baseline model (SiMVC) that aligned the distributions of the views. Further, it added the contrastive module and selective views alignment by prioritizing the views and, thus, improved the baseline model's performance calling it as CoMVC framework [172]. Therefore, the experiments were conducted with the CoMVC part only that gave the best results.

4.4 Results and Analysis

It is generally presumed that that the quantity of clusters is already established while conducting experiments. In such cases, various metrics, including accuracy, Normalized Mutual Information (NMI), and Adjusted Rand Index (ARI), are commonly employed [173, 174]. Thus, the evaluation of the proposed model's performance was carried out using these three metrics.

Some of the metrics are described below:

- Normalized Mutual Information (NMI): This metric computes the normalized measure of similarity between the labels of same data instances. The range of NMI is $[0, 1]$ where 0 signifies no correlation and 1 signifies the perfect correlation. The formula is given by:

$$NMI = \frac{I(l, c)}{max(H(l), H(c))} \tag{4.3}$$

  where I(l,c) denotes the mutual information between the true label l and the assigned cluster c and H denotes the entropy.

- Adjusted Rand Index (ARI): ARI measures the similarity between two clusters by considering all pairs of data instances that are assigned to the same or different clusters in the actual and predicted labels. The range of ARI is $[-1, 1]$. The higher the ARI value, the better is the clustering.

$$ARI = \frac{(RI - E)}{(max(RI) - E)} \tag{4.4}$$

where RI = Rand Index and E is the Expected Rand Index Value for random clusterings. These are:

$$RI = \frac{(a+b)}{\binom{n}{2}} \tag{4.5}$$

where a = the number of times a pair of elements belongs to the same cluster across two clustering methods, b = the number of times a pair of elements belong to different clusters across two clustering methods and $\binom{n}{2}$ is the number of unordered pairs in a set of n elements. Here, max(RI) = 1.

and

$$E = (\sum(\binom{n_i}{2}) \times \sum(\binom{n_j}{2}))/(\binom{N}{2}) \tag{4.6}$$

where n_i is the number of samples in cluster i and n_j is the number of samples in cluster j.

The results of the proposed model and benchmarks on all four datasets are reported in Table 4.2. It can be observed from Table 4.2 that for all datasets, the proposed model has shown better performance than the state-of-the-arts except for NMI values for Mfeat and ALOI datasets and ARI in the case of ALOI. It is worth noting that the proposed technique performed well in the case of WebKB and ALOI, both of which had fewer samples than the number of clusters to be identified. In the case of Mfeat and ALOI, it reached good Accuracy and ARI values for Mfeat. Thus, the proposed method performed well for challenging datasets and slightly worse for easier ones. The overall performance of the proposed approach was better than the state-of-the-arts.

Also, the convergence plot for all the datasets were plotted that can be referred

Table 4.2: Clustering Results. All the metrics in (%)

Models	100leaves			WebKB			Mfeat			ALOI		
	Acc	**NMI**	**ARI**	**Acc**	**NMI**	**ARI**	**Acc**	**NMI**	**ARI**	**Acc**	**NMI**	**ARI**
MCGL	81.06	91.30	51.50	54.19	8.60	4.01	85.30	**90.55**	83.13	46.25	66.57	4.41
GMC	82.38	92.92	49.74	76.35	41.64	42.80	88.20	90.50	85.02	57.05	73.50	43.05
DEMVC	6.69	24.53	0.60	49.75	10.05	8.43	46.45	37.53	24.59	13.52	41.30	8.45
RRA-MVC	73.25	92.56	71.58	40.89	13.43	9.22	81.20	83.19	74.36	55.22	**80.79**	**49.34**
Proposed	**91.13**	**96.59**	**88.01**	**80.79**	**54.98**	**52.02**	**95.00**	89.22	**89.89**	**58.95**	79.75	46.84

to from Fig. 4.3. Using SGD as an optimizer, it could be clearly inferred that the given solution converged to the point of stability. The SGD parameters, such as mini-batch size and learning rate, are given in Table 4.1 for all the considered datasets.

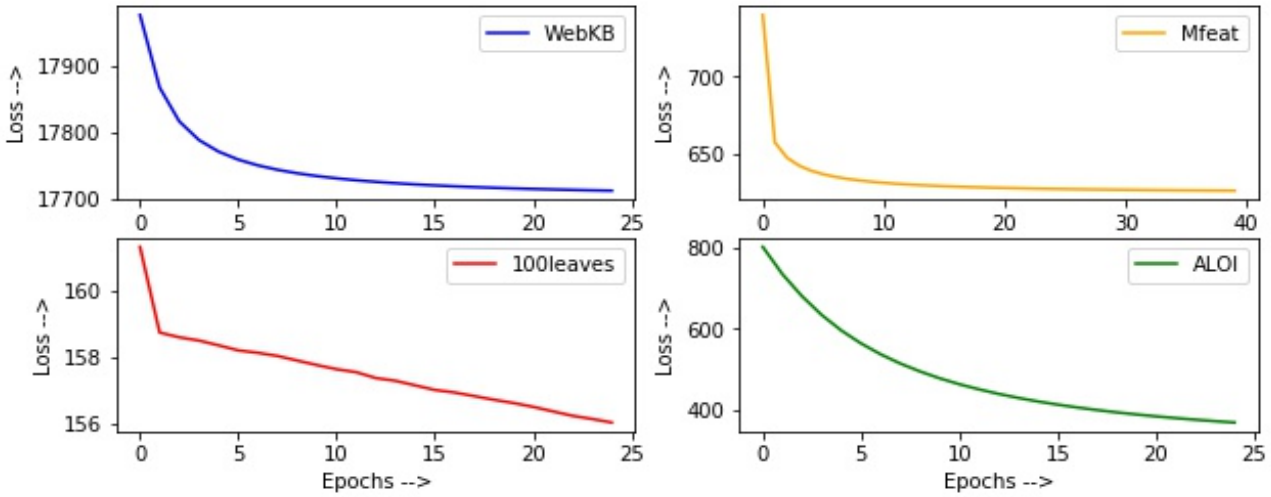


Figure 4.3: Loss Plots

4.4.1 Ablation studies

This section shows the results corresponding to the three ablation studies performed for all the datasets. The first experiment conducted was with changing the values of the regularizers λ and μ associated with the penalty terms log-det and Frobenius norms in both CTL and TL equations 2.11 and 2.14 respectively.

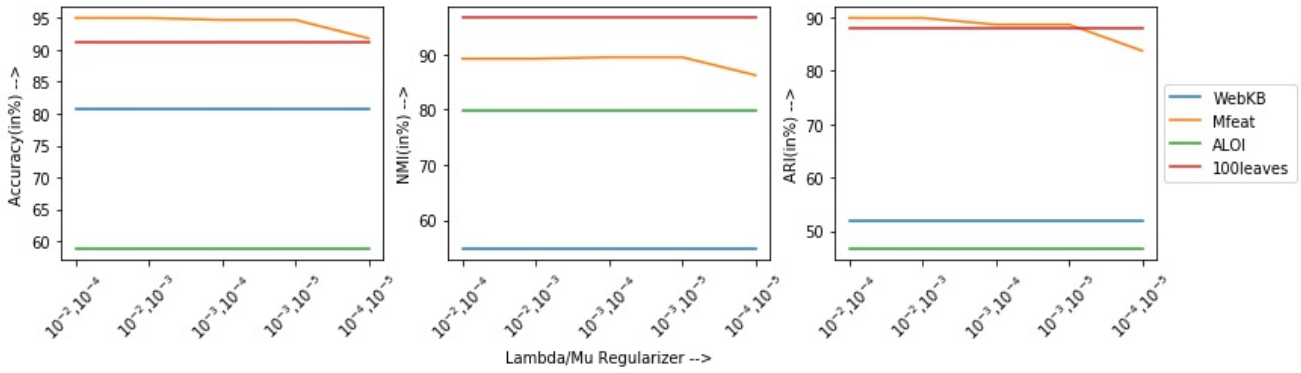


Figure 4.4: Ablation Studies Result Plots on λ, μ

The set of values taken as a combination for both the penalty regularizers are - $((10^{-2}, 10^{-4}), (10^{-2}, 10^{-3}), (10^{-3}, 10^{-4}), (10^{-3}, 10^{-5}),$ $(10^{-4}, 10^{-5}))$. The results can be referred from Table 4.3. These were also displayed graphically for all three metrics Accuracy, NMI and ARI in Fig. 4.4. It can be clearly concluded from the results that the penalization terms play an essential role in our formulation. Although, while changing the values of regularizers for these penalizations, the change is robust for three of the datasets used from the performance evaluation perspective. However, the depleted results for Mfeat for lower values of these regularizers demonstrate that they help to learn better representations and hence should be the part of the formulation.

Table 4.3: Ablation Studies Results on λ, μ

Value	**100leaves** ($\beta = 1.0$)			**WebKB** ($\beta = 0.5$)			**Mfeat** ($\beta = 0.8$)			**ALOI** ($\beta = 0.5$)		
(λ, μ)	**Acc**	**NMI**	**ARI**	**Acc**	**NMI**	**ARI**	**Acc**	**NMI**	**ARI**	**Acc**	**NMI**	**ARI**
$(10^{-2}, 10^{-4})$	91.13	96.59	88.01	80.79	54.98	52.02	95.00	89.22	89.89	58.95	79.75	46.84
$(10^{-2}, 10^{-3})$	91.13	96.59	88.01	80.79	54.98	52.02	95.00	89.22	89.89	58.95	79.75	46.84
$(10^{-3}, 10^{-4})$	91.13	96.59	88.01	80.79	54.98	52.02	94.70	89.49	88.66	58.95	79.75	46.84
$(10^{-3}, 10^{-5})$	91.13	96.59	88.01	80.79	54.98	52.02	94.70	89.49	88.66	58.95	79.75	46.84
$(10^{-4}, 10^{-5})$	91.13	96.59	88.01	80.79	54.98	52.02	91.80	86.24	83.74	58.95	79.75	46.84

Secondly, the experiments were carried out with the regularizer β associated with K-Means clustering loss in the equation 4.2. The set of values for β lie in

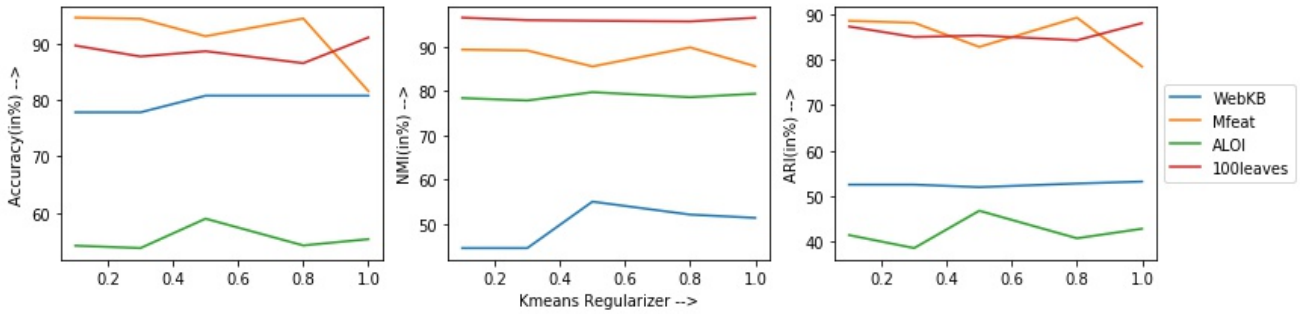


Figure 4.5: Ablation Studies Result Plots on K-Means Regularizer

range $[0, 1]$, specifically, these are $(0.0, 0.1, 0.3, 0.5, 0.8, 1.0)$. The results were represented results both in text and graphically that can be referred from Table 4.4 and Fig. 4.5 respectively. It could be observed that for all the datasets, in general, the K-Means regularizer $\beta \geq 0.5$ gave better performance. This signified that K-Means loss was an important term associated with the final objective function. It helped in learning better representations as guided by it and was thus responsible for better clustering performance.

Table 4.4: Ablation Studies Results on K-Means Regularizer

Value	**100leaves**			**WebKB**			**Mfeat**			**ALOI**		
β	**Acc**	**NMI**	**ARI**	**Acc**	**NMI**	**ARI**	**Acc**	**NMI**	**ARI**	**Acc**	**NMI**	**ARI**
0.0	89.56	96.17	86.50	77.83	44.47	52.57	91.10	85.37	82.15	55.27	78.34	41.16
0.1	89.69	**96.62**	87.26	77.83	44.47	52.57	94.65	89.39	88.51	54.15	78.41	41.49
0.3	87.75	96.05	84.96	77.83	44.47	52.57	94.45	89.18	88.08	53.72	77.87	38.65
0.5	88.69	95.91	85.30	**80.79**	**54.98**	52.02	91.35	85.57	82.79	**58.95**	**79.75**	**46.84**
0.8	86.56	95.76	84.24	80.79	52.05	52.81	**95.00**	**89.22**	**89.89**	54.20	78.61	40.80
1.0	**91.13**	96.59	**88.01**	80.79	51.32	**53.24**	81.60	85.62	78.46	55.31	79.40	42.89

The second experiment inference was also validated by the third experiment conducted, where the results were computed using piecemeal version of the proposed model. It means that first the learned representations from the DeConFuse network were learned separately and subsequently passed these representations via the K-Means clustering module to get the final clusters, i.e., here $\beta = 0$. The

results could be referred from Table 4.5. Here, it was clearly inferred that the joint optimization of the DeConFuse and K-Means clustering module is better than the piecemeal approach.

Table 4.5: Ablation Studies Results on Piecemeal and Proposed Formulation

Methods	**100leaves**			**WebKB**			**Mfeat**			**ALOI**		
	Acc	**NMI**	**ARI**	**Acc**	**NMI**	**ARI**	**Acc**	**NMI**	**ARI**	**Acc**	**NMI**	**ARI**
Piecemeal	89.56	96.17	86.50	77.83	44.47	**52.57**	91.10	85.37	82.15	55.27	78.34	41.16
Proposed	**91.13**	**96.59**	**88.01**	**80.79**	**54.98**	52.02	**95.00**	**89.22**	**89.89**	**58.95**	**79.75**	**46.84**

4.5 Discussion

In this chapter, a novel unsupervised multi-channel fusion clustering framework based on Deep Convolutional Transform Learning named DeConFCluster is discussed. The proposed framework jointly trains the DCTL based DeConFuse and K-Means clustering modules in an end-to-end fashion. The advantage of this framework is that it does not have the additional overhead of learning the weights of decoder or deconvolutional layers, which is the case in existing multiview clustering approaches. Secondly, the framework avoided overfitting even in data-constrained scenarios where the number of data instances is low and the number of classes is high, for example, in the case of 100leaves and WebKB, the proposed framework performed well compared to benchmarks.

Another advantage of this framework is that it promotes diversity among filters and thus, in turn, helps to learn more interpretable filters that are further guided by K-Means loss. Therefore, due to these advantages, the proposed framework DeConFCluster, evaluated on the four standard multiview datasets,

demonstrated higher clustering scores as compared to the current state-of-the-art MVC frameworks. However, for a few metrics in the case of Mfeat and ALOI datasets, the performance of the method needs to be improved which can be worked upon in the future.

Chapter 5

Conclusion

The proposed works in this thesis focused on modeling various prediction problems in the field of Information Fusion as multi-channel fusion problems. The frameworks proposed are based on the recently established technique CTL and hence are variants that deal in the analysis domain, covering both unsupervised and supervised learning paradigms.

5.1 Summary of Contribution

In this section, the chapter-wise contributions are briefly summarized in the area of Information Fusion, giving a bird's eye view of the dissertation.

5.1.1 Unsupervised multi-channel CTL based fusion frameworks - ConFuse(shallow) and DeConFuse(Deep)

In this part of the dissertation, an unsupervised multi-channel fusion framework is modeled as both shallow and deep architectures based on CTL, namely - ConFuse

and DeConFuse, respectively. These frameworks are applied to the problem of Stock trading (classification) and forecasting (regression) and obtained good performance successfully. Since the data is time-series, the framework treated the data as univariate versus 2D matrix/image, as discussed in that chapter. It also guaranteed distinct filters and produced more interpretable representations. The most significant advantage of this framework was that it avoided the effort of re-training the network which is required with most other techniques, especially CNNs. In summary, the same representations were utilized for both regression and classification tasks without requiring them to be learned separately through different trainings.

5.1.2 Supervised multi-channel fusion frameworks - SuperDeConFuse and DeConDFFuse

This chapter presented the supervised multi-channel fusion frameworks based on CTL. The first framework involved fusion that happened via TL over the representations learned from individual CTL based channels. Further, there was a linear fully connected layer followed by the cross-entropy loss. The framework is called as - SuperDeConFuse (SDCF). It is applied to the Stock trading data. The performance of the proposed technique showed that the proposed method worked well versus the state-of-the-art. The non-negativity constraint on the fused representation learned, i.e., Z helped to eliminate the problem of dead neurons and did not require us to employ activation function between the last convolutional layer and fully connected layer, i.e., fusion layer learned via TL versus the required in CNNs. All the other advantages, like distinct filters and

interpretable representations, etc., are also present here since they are based on CTL and TL. It has been even compared with the representations learned from CNNs and found that the features and results from the proposed frameworks are better than those from CNNs.

The other framework that is discussed in this chapter is DeConDFFuse which extended the DeConFuse network and jointly trained and optimized it with Decision Forest (DF). Again, all the benefits of CTL are applicable here as well. The representations/features learned are guided by DF apart from CTL. The framework is applied to the Drug-Drug Interaction (DDI) problem that predicts two kinds of interactions - known-to-interact (1) and known-no-to-interact or unknown solving a fundamental issue to prevent Adverse Drug Reactions (ADRs). The contribution of this model is that it utilizes DF in a joint framework which is generally used in a piecemeal fashion. The chances of missing any important information are there in the latter. Additionally, it extracted individual and cross-channel features of the drugs finding out the most relevant features of the drugs that interact with each other. The results from this framework are superior to benchmarks indicating the benefit of employing it for the DDI prediction task.

5.1.3 Multiview Clustering Framework based on CTL - DeConFCluster

Lastly, multiview clustering performing framework - DeConFCluster is explained in chapter 4. It is an unsupervised multiview multi-channel fusion framework that performs multiview clustering task utilizing representations from the fusion of the

individual view's representations; thus, it learns individual view information and then learns cross-channel information via fusion. The framework comprised a DeConFuse network and a K-means module that are jointly trained and optimized. Therefore, representations learned are beneficial since those have the advantages of CTL and are well-guided through the K-Means loss also. The same is observed in terms of performance from the experimental results too. Furthermore, the framework prevented the additional training from the decoder network that is generally applied in the case of multiview clustering approaches. Besides the said advantage, the framework also avoided overfitting even in data-constrained scenarios where the number of data instances is low and the number of classes is high.

5.2 Future Work

It is believed that the algorithms proposed are generic and can be used not only in the kind of problems discussed throughout this dissertation but also in other research fields where one can formulate the problem to be solved as a multi-channel fusion problem. Both supervised and unsupervised frameworks have been proposed; thus, the proposed frameworks can solve both genres' problems.

At the application level, the unsupervised and supervised frameworks were applied for stock prediction problems that required day-wise predictions. However, in future, it is desired to experiment with the proposed frameworks at more micro-level predictions in stock, i.e., at minute-level prediction of signals - BUY

and SELL. Also, the proposed solutions have dealt with 1D data so far and, thus, used 1D Convolutions. Therefore, it encourages to explore these frameworks to be applied to problems that involve 2D or multidimensional data. For example, a hyperspectral image and an RGB image can be used to perform fusion to estimate dense depth maps from the sparse maps. Thus, such a fusion using these frameworks find their application in drones.

Likewise, the supervised learning framework - DeConDFFuse is utilized for DDI prediction, but it is believed that it can also be used to predict different types of associations in bio-informatics like - drug-virus, drug-target, protein-protein, etc. Also, the problem targeted currently involves two drugs administered together, whereas many times, more than two are administered together in a real scenario. The latter can be easily done with the proposed network.

Similarly, in this thesis, unsupervised frameworks - ConFuse and DeConFuse- are developed to extract features for performing dual tasks of regression and classification. However, apart from these tasks, it is worthy to analyze if such features from the framework can be successfully utilized in other applications as well, like anomaly detection. Anomaly detection requires finding and identifying outliers to prevent fraud, adversary attacks, network intrusions, etc., that can compromise any organization's future or invades an individual's privacy. Further, it can be even utilized to extract such features to find clusters to segment customers pertaining to a particular market. In all such applications, one can analyze the data available and extract meaningful representations to perform the mentioned tasks using the proposed framework. Also, unsupervised (ConFuse and

DeConFuse) and supervised frameworks (SuperDeConFuse and DeConDFFuse) can be employed in another application of Human Activity Recognition.

Additionally, the multiview clustering framework performs clustering on views that are similar in nature. Nevertheless, it can even be considered extending this framework to apply to multi-modal datasets involving multiple modalities like image, text, video, and audio information. For example, Caltech-UCSD Birds-200-2011, shortly called as CUB-200-2011 dataset, contains image and text information.

Lastly, it is intended to implement techniques following a semi-supervised learning paradigm. Semi-supervised learning based formulations will hold immense importance as these will help perform tasks involving partially labeled data. It basically reduces expenses on manual annotation and cuts down on data preparation time which is significant as unlabeled data is available in abundance.

References

[1] P. Kontschieder, M. Fiterau, A. Criminisi, and S. R. Bulò, "Deep neural decision forests," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1467–1475.

[2] H. Boström, S. F. Andler, M. Brohede, R. Johansson, A. Karlsson, J. van Laere, L. Niklasson, M. Nilsson, A. S. Persson, and T. Ziemke, "On the definition of information fusion as a field of research," 2007.

[3] Y. Yoon, J. Cho, and G. Yoon, "Non-constrained blood pressure monitoring using ecg and ppg for personal healthcare," *Journal of medical systems*, vol. 33(4), pp. 261–266, 2009.

[4] N.-E. El Faouzi, H. Leung, and A. Kurian, "Data fusion in intelligent transportation systems: Progress and challenges – a survey," *Information Fusion*, vol. 12, pp. 4–10, 01 2011.

[5] I. Saadi, B. Farooq, A. Mustafa, J. Teller, and M. Cools, "An efficient hierarchical model for multi-source information fusion," *Expert Systems with Applications*, vol. 110, pp. 352 – 362, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417418303646

[6] F. Rodrigues, I. Markou, and F. C. Pereira, "Combining time-series and textual data for taxi demand prediction in event areas: A deep learning approach," *Information Fusion*, vol. 49, pp. 120 – 129, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1566253517308175

[7] S. Daneshvar and H. Ghassemian, "Mri and pet image fusion by combining ihs and retina-inspired models," *Information Fusion*, vol. 11(2), pp. 114–123, 2010.

[8] T. D. Dixon, S. G. Nikolov, J. J. Lewis, J. Li, E. F. Canga, J. M. Noyes, T. Troscianko, R. D. Bull, and C. N. Canagarajah, "Task-based scanpath assessment of multi-sensor video fusion in complex scenarios," *Information Fusion, Special Issue on Biologically-Inspired Information Fusion*, vol. 11(1), pp. 51–65, 2010.

[9] J. A. Balazs and J. D. Velásquez, "Opinion mining and information fusion: A survey," *Information Fusion*, vol. 27, pp. 95–110, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253515000536

[10] L. Sun, W. Xu, and J. Liu, "Two-channel attention mechanism fusion model of stock price prediction based on cnn-lstm," *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 20, no. 5, jul 2021. [Online]. Available: https://doi.org/10.1145/3453693

[11] S. Lin, Y. Wang, L. Zhang, Y. Chu, Y. Liu, Y. Fang, M. Jiang, Q. Wang,

B. Zhao, Y. Xiong, and D.-Q. Wei, "MDF-SA-DDI: predicting drug–drug interaction events based on multi-source drug fusion, multi-source feature fusion and transformer self-attention mechanism," *Briefings in Bioinformatics*, vol. 23, no. 1, 10 2021, bbab421. [Online]. Available: https://doi.org/10.1093/bib/bbab421

[12] J. Yang, M. Nguyen, P. San, X. Li, and S. Krishnaswamy, "Deep convolutional neural networks on multichannel time series for human activity recognition," *In Twenty-Fourth International Joint Conference on Artificial Intelligence (June 2015)*, June 2015.

[13] T. Meng, X. Jing, Z. Yan, and W. Pedrycz, "A survey on machine learning for data fusion," *Information Fusion*, vol. 57, pp. 115–129, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253519303902

[14] C. Ounoughi and S. Ben Yahia, "Data fusion for its: A systematic literature review," *Information Fusion*, vol. 89, pp. 267–291, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253522001087

[15] I. Belhajem, Y. Ben Maissa, and A. Tamtaoui, "A robust low cost approach for real time car positioning in a smart city using extended kalman filter and evolutionary machine learning," in *2016 4th IEEE International Colloquium on Information Science and Technology (CiSt)*, 2016, pp. 806–811.

[16] I. Belhajem, Y. M. Ben, and A. Tamtaoui, "Improving vehicle localization in a smart city with low cost sensor networks and support vector machines," *Mobile Networks and Applications*, vol. 23, pp. 1–10, 08 2018.

[17] G. Bresson, R. Aufrère, and R. Chapuis, "A general consistent decentralized simultaneous localization and mapping solution," *Robotics and Autonomous Systems*, vol. 74, pp. 128–147, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889015001529

[18] H. Cho, Y.-W. Seo, B. V. Kumar, and R. R. Rajkumar, "A multi-sensor fusion system for moving object detection and tracking in urban driving environments," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 1836–1843.

[19] K. Golestan, S. Seifzadeh, M. Kamel, F. Karray, and F. Sattar, "Vehicle localization in vanets using data fusion and v2v communication," in *Proceedings of the Second ACM International Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications*, ser. DIVANet '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 123–130. [Online]. Available: https://doi.org/10.1145/2386958.2386977

[20] K. Golestan, F. Sattar, F. Karray, M. S. Kamel, and S. Saifzadeh, "Localization in vehicular ad hoc networks using data fusion and v2v communication," *Computer Communications*, vol. 71, 07 2015.

[21] A. Vu, A. Ramanandan, A. Chen, J. A. Farrell, and M. Barth, "Real-time computer vision/dgps-aided inertial navigation system for lane-level vehi-

cle navigation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 2, pp. 899–913, 2012.

[22] K. Lassoued, I. Fantoni, and P. Bonnifait, "Mutual localization and positioning of vehicles sharing gnss pseudoranges: Sequential bayesian approach and experiments," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, 2015, pp. 1896–1901.

[23] C. Sutton, C. Morrison, P. Cohen, J. Moody, and J. Adibi, "A bayesian blackboard for information fusion," *Proceedings of the Seventh International Conference on Information Fusion, FUSION 2004*, vol. 2, 07 2004.

[24] Q. Miao, Q. Li, and D. Zeng, "Mining fine grained opinions by using probabilistic models and domain knowledge," in *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, vol. 1, 2010, pp. 358–365.

[25] G. Shroff, P. Agarwal, and L. Dey, "Enterprise information fusion for real-time business intelligence," in *14th International Conference on Information Fusion*, 2011, pp. 1–8.

[26] T. Rohe, A.-C. Ehlis, and U. Noppeney, "The neural dynamics of hierarchical bayesian causal inference in multisensory perception," *Nature Communications*, vol. 10(1), pp. 1–17, 2019.

[27] N. Nesa and I. Banerjee, "Iot-based sensor data fusion for occupancy

sensing using dempster–shafer evidence theory for smart buildings," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1563–1570, 2017.

[28] W. Ding, X. Jing, Z. Yan, and L. T. Yang, "A survey on data fusion in internet of things: Towards secure and privacy-preserving fusion," *Information Fusion*, vol. 51, pp. 129–144, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253518304731

[29] S.-H. Chen, J.-S. Pan, K. Lu, and H. Xu, "Driving behavior analysis of multiple information fusion based on adaboost," *Advances in Intelligent Systems and Computing*, vol. 329, pp. 277–285, 01 2015.

[30] A. Ben Mahjoub, M. Ibn Khedher, M. Atri, and M. El Yacoubi, "Naive bayesian fusion for action recognition from kinect," in *4th International Conference on Computer Networks & Data Communications*, 12 2017.

[31] X. Sevillano, E. Màrmol, and V. Fernandez-Arguedas, "Towards smart traffic management systems: Vacant on-street parking spot detection based on video analytics," in *17th International Conference on Information Fusion (FUSION)*, 2014, pp. 1–8.

[32] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? sentiment classification using machine learning techniques," in *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*. Association for Computational Linguistics, Jul. 2002, pp. 79–86. [Online]. Available: https://aclanthology.org/W02-1011

[33] F. Li, Y. Fan, X. Zhang, C. Wang, F. Hu, W. Jia, and H. Hui, "Multi-

feature fusion method based on eeg signal and its application in stroke classification," in *Journal of medical systems*, vol. 44(2). Association for Computational Linguistics, 2019, p. 39.

[34] T. P. Banerjee and S. Das, "Multi-sensor data fusion using support vector machine for motor fault detection," *Information Sciences*, vol. 217, pp. 96–107, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025512004185

[35] Z. Yao and W. Yi, "License plate detection based on multistage information fusion," *Information Fusion*, vol. 18, pp. 78–85, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253513000663

[36] B. Weng, M. A. Ahmed, and F. M. Megahed, "Stock market one-day ahead movement prediction using disparate data sources," *Expert Systems with Applications*, vol. 79, pp. 153–163, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417417301331

[37] X. Chen, H. Xie, Z. Li, G. Cheng, M. Leng, and F. L. Wang, "Information fusion and artificial intelligence for smart healthcare: a bibliometric study," *Information Processing & Management*, vol. 60, no. 1, p. 103113, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S030645732200214X

[38] Y. Yang, J. Wu, S. Huang, Y. Fang, P. Lin, and Y. Que, "Multimodal medical image fusion based on fuzzy discrimination with structural patch

decomposition," *IEEE Journal of Biomedical and Health Informatics*, vol. 23, no. 4, pp. 1647–1660, 2019.

[39] F. Wang, K. Wang, and F. Jiang, "An improved fusion method of fuzzy logic based on k-mean clustering in wsn," *Sensors and Transducers*, vol. 157, pp. 20–25, 01 2013.

[40] S. Xiao, Y. Zhang, X. Liu, and J. Gao, "Alert fusion based on cluster and correlation analysis," in *Proceedings of the 2008 International Conference on Convergence and Hybrid Information Technology*, ser. ICHIT '08. USA: IEEE Computer Society, 2008, p. 163–168. [Online]. Available: https://doi.org/10.1109/ICHIT.2008.197

[41] A. Rodriguez-Castaño, G. Heredia, and A. Ollero, "High-speed autonomous navigation system for heavy vehicles," *Applied Soft Computing*, vol. 43, pp. 572–582, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494616300771

[42] K. Saadeddin, M. F. Abdel-Hafez, M. A. Jaradat, and M. A. Jarrah, "Performance enhancement of low-cost, high-accuracy, state estimation for vehicle collision prevention system using anfis," *Mechanical Systems and Signal Processing*, vol. 41, no. 1, pp. 239–253, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0888327013002963

[43] H. Medjahed, D. Istrate, J. Boudy, J.-L. Baldinger, and B. Dorizzi, "A pervasive multi-sensor data fusion for smart home healthcare monitoring,"

in *2011 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2011)*, 2011, pp. 1466–1473.

[44] A. Oztekin, R. Kizilaslan, S. Freund, and A. Iseri, "A data analytic approach to forecasting daily stock returns in an emerging market," *European Journal of Operational Research*, vol. 253, no. 3, pp. 697–710, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221716301096

[45] S. Yao, S. Hu, Y. Zhao, A. Zhang, and T. Abdelzaher, "Deepsense: A unified deep learning framework for time-series mobile sensing data processing," *In Proceedings of the 26th International Conference on World Wide Web*, pp. 351–360, April 2017.

[46] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. Zhao, "Time series classification using multi-channels deep convolutional neural networks," *In International Conference on Web-Age Information Management,Springer, Cham*, pp. 289–310, June 2014.

[47] B. Pu, Y. Liu, N. Zhu, K. Li, and K. Li, "Ed-acnn: Novel attention convolutional neural network based on encoder–decoder framework for human traffic prediction," *Applied Soft Computing*, vol. 97, p. 106688, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494620306268

[48] R. Dheekonda, S. Panda, M. Khan, M. Hasan, and S. Anwar, "Object detection from a vehicle using deep learning network and future integration

with multi-sensor fusion algorithm," in WCX^{TM} *17 SAE World Congress*, 03 2017.

[49] G. Giacinto, F. Roli, and L. Didaci, "Fusion of multiple classifiers for intrusion detection in computer networks," *Pattern Recognition Letters*, vol. 24, no. 12, pp. 1795–1803, 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865503000047

[50] X. Chen, J. Chen, G. Cheng, and T. Gong, "Topics and trends in artificial intelligence assisted human brain research," *PLoS ONE*, vol. 15, no. 4, 2020.

[51] S. Hashemi, H. Veisi, E. Jafarzadehpour, R. Rahmani, and Z. Heshmati, "Multi-view deep learning for rigid gas permeable lens base curve fitting based on pentacam images," *Medical & Biological Engineering & Computing*, vol. 58, 05 2020.

[52] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1993–1941, 2016.

[53] A. Eitel, J. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard, "Multimodal deep learning for robust rgb-d object recognition," *In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 681–687, September 2015.

[54] Y. Chen, C. Li, P. Ghamisi, X. Jia, and Y. Gu, "Deep fusion of remote

sensing data for accurate classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 14(8), pp. 1253–1257, 2017.

[55] N. Antropova, B. Huynh, and M. Giger, "A deep feature fusion methodology for breast cancer diagnosis demonstrated on three imaging modality datasets," *Medical physics*, vol. 44(10), pp. 5162–5171, 2017.

[56] J. Maggu, E. Chouzenoux, G. Chierchia, and A. Majumdar, "Convolutional transform learning," *In International Conference on Neural Information Processing*, pp. 162–174, Dec 2018.

[57] P. Gupta, A. Majumdar, E. Chouzenoux, and G. Chierchia, "Superdeconfuse: A supervised deep convolutional transform based fusion framework for financial trading systems," *Expert Systems with Applications*, vol. 169, p. 114206, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417420309349

[58] M. Bertoni, M. Duran-Frigola, P. Badia-i Mompel, E. Pauls, M. Orozco-Ruiz, O. Guitart-Pla, V. Alcalde, V. M. Diaz, A. Berenguer-Llergo, I. Brun-Heath, N. Villegas, A. G. de Herreros, and P. Aloy, "Bioactivity descriptors for uncharacterized chemical compounds," *Nature Communications*, no. 3932, 2021.

[59] H. Wang, Y. Yang, and B. Liu, "Gmc: Graph-based multi-view clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 6, pp. 1116–1129, 2019.

[60] X. Zhang, L. Zhao, L. Zong, X. Liu, and H. Yu, "Multi-view clustering

via multi-manifold regularized nonnegative matrix factorization," in *Proceedings of the IEEE International Conference on Data Mining (ICDM 2014)*, 2014, pp. 1103–1108.

[61] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," in *In Competition and cooperation in neural nets*, 1982, p. 267–285.

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

[63] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012, pp. 84–90. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[64] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[65] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE

Computer Society, jun 2015, pp. 1–9. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR.2015.7298594

[66] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[67] J. Maggu, A. Majumdar, E. Chouzenoux, and G. Chierchia, "Deep convolutional transform learning," in *Proceedings of the International Conference on Neural Information Processing (ICONIP 2000)*, 2020, pp. 300–307.

[68] P. Gupta, J. Maggu, A. Majumdar, E. Chouzenoux, and G. Chierchia, "Confuse: Convolutional transform learning fusion framework for multi-channel data analysis," in *2020 28th European Signal Processing Conference (EUSIPCO)*, 2021, pp. 1986–1990.

[69] P. Gupta, J. Maggu, E. Majumdar, A. Chouzenoux, and G. Chierchia, "Deconfuse: a deep convolutional transform-based unsupervised fusion framework," *EURASIP J. Adv. Signal Process*, vol. 26, 2020.

[70] P. Gupta, A. Majumdar, E. Chouzenoux, and G. Chierchia, "Decondffuse : Predicting drug–drug interaction using joint deep convolutional transform learning and decision forest fusion framework," *Expert Systems with Applications*, vol. 227, p. 120238, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417423007406

[71] P. Gupta, A. Goel, A. Majumdar, E. Chouzenoux, and G. Chierchia,

"Deconfcluster: Deep convolutional transform learning based multiview clustering fusion framework," *Submitted in IEEE TNNLS*, 2023.

[72] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data Mining and Knowledge Discovery*, vol. 33, pp. 917–963, 07 2019.

[73] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, nov 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[74] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Gated feedback recurrent neural networks," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. JMLR.org, 2015, p. 2067–2075.

[75] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *International Joint Conference on Neural Networks (IJCNN)*, 05 2017, pp. 1578–1585.

[76] P. Malhotra, V. Tv, L. Vig, P. Agarwal, and G. Shroff, "Timenet: Pretrained deep recurrent neural network for time series classification," in *25th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 04 2017.

[77] [Online]. Available: https://www.cs.ucr.edu/~eamonn/time_series_data/

[78] K. Kashiparekh, J. Narwariya, P. Malhotra, L. Vig, and G. Shroff, "Convtimenet: A pre-trained deep convolutional neural network for time series

classification," in *International Joint Conference on Neural Networks (IJCNN)*, 07 2019, pp. 1–8.

[79] J. Debayle, N. Hatami, and Y. Gavet, "Classification of time-series images using deep convolutional neural networks," in *Tenth International Conference on Machine Vision (ICMV*, 04 2018, p. 23.

[80] Z. Wang and T. Oates, "Imaging time-series to improve classification and imputation," in *Proceedings of the 24th International Conference on Artificial Intelligence*, ser. IJCAI'15. AAAI Press, 2015, p. 3939–3945.

[81] O. Sezer and M. Ozbayoglu, "Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach," *Applied Soft Computing*, vol. 70, 04 2018.

[82] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Forecasting stock prices from the limit order book using convolutional neural networks," in *2017 IEEE 19th Conference on Business Informatics (CBI)*, vol. 01, 2017, pp. 7–12.

[83] M. U. Gudelek, S. A. Boluk, and A. M. Ozbayoglu, "A deep learning based stock trading model with 2-d cnn trend detection," *In 2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8, November 2017.

[84] S. Ravishankar and Y. Bresler, "Sparsifying transform learning with efficient optimal updates and convergence guarantees," *IEEE Transactions on Signal Processing*, vol. 63, no. 9, pp. 2389–2404, 2015.

[85] H. Bauschke, R. Burachik, P. Combettes, and D. Luke, *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*. Springer Optimization and Its Applications, 11 2009.

[86] J. Sun, Y. Qing, C. Liu, and J. Lin, "Self-fts: A self-supervised learning method for financial time series representation in stock intraday trading," in *2022 IEEE 20th International Conference on Industrial Informatics (INDIN)*, 2022, pp. 501–506.

[87] Y. Soun, J. Yoo, M. Cho, J. Jeon, and U. Kang, "Accurate stock movement prediction with self-supervised learning from sparse noisy tweets," in *2022 IEEE International Conference on Big Data (Big Data)*, 2022, pp. 1691–1700.

[88] Y. Yang and T. M. Hospedales, "An evaluation of self-supervised learning for portfolio diversification," in *International Conference on Artificial Neural Networks (ICANN)*, 2022, pp. 1691–1700.

[89] K. Xu, G. Zhong, Z. Deng, K. Zhang., and H. K., "Self-supervised generative learning for sequential data prediction," in *International Conference on Artificial Neural Networks (ICANN)*, 2023.

[90] S. Ravishankar and Y. Bresler, "Learning sparsifying transforms," *IEEE Transactions on Signal Processing*, vol. 61(5), pp. 1072–1086, 2012.

[91] H. Attouch, J. Bolte, and B. Svaiter, "Convergence of descent methods for semi-algebraic and tame problems: Proximal algorithms, forward-

backward splitting, and regularized gauss-seidel methods," *Mathematical Programming*, vol. 137, 01 2011.

[92] E. Chouzenoux, J.-C. Pesquet, and A. Repetti, "A block coordinate variable metric forward-backward algorithm," *Journal of Global Optimization*, vol. 66, 11 2016.

[93] J. Bolte, S. Sabach, and M. Teboulle, "Proximal alternating linearized minimization for nonconvex and nonsmooth problems," *Mathematical Programming*, vol. 146, 08 2013.

[94] P. Combettes and J.-C. Pesquet, "Deep neural network structures solving variational inequalities," *Set-Valued and Variational Analysis*, 2020, https://arxiv.org/abs/1808.07526.

[95] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. of ICLR*, 2015.

[96] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Proc. of NeurIPS*, Long Beach, California, USA, 4-9 Dec. 2017.

[97] A. Mass, A. Hannun, and A. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. of ICML*, Atlanta, USA, 16-21 June 2013.

[98] S. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," *Proc. ICLR.*, 2018.

[99] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," *NIPS Autodiff Workshop*, 2017.

[100] C. Kocak, "Arma(p,q) type high order fuzzy time series forecast method based on fuzzy logic relations," *Applied Soft Computing*, vol. 58, pp. 92–103, 2017.

[101] G. Zumbach and L. Fernndez, "Option pricing with realistic arch processes," *Quantitative Finance*, vol. 14(1), pp. 143–170, 2014.

[102] Z. Lin, "Modelling and forecasting the stock market volatility of sse composite index using garch models," *Future Generation Computer Systems*, vol. 79, pp. 960–972, 2018.

[103] N. lk, D. Kuruppuarachchi, and O. Kuzmicheva, "Stock market's response to real output shocks in eastern european frontier markets: A varwal model," *Emerging Market Review*, vol. 33, pp. 140 – 154, 2017.

[104] R. Bisoi and P. Dash, "A hybrid evolutionary dynamic neural network for stock market trend analysis and prediction using unscented kalman filter," *Applied Soft Computing*, vol. 19, pp. 41–56, 2014.

[105] O. B. Sezer and A. M. Ozbayogl, "Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach," *Applied Soft Computing*, vol. 70, pp. 525–538, 2018.

[106] F. Ming, F. Wong, Z. Liu, and M. Chiang, "Stock market prediction from

wsj: Text mining via sparse matrix factorization," *2014 IEEE International Conference on Data Mining, Shenzhen*, pp. 430–439, 2014.

[107] Y. Shynkevich, T. McGinnity, S. Coleman, A. Belatreche, and Y. Li, "Forecasting price movements using technical indicators: investigating the impact of varying input window length," *Neurocomputing*, vol. 164, pp. 163–173, 2017.

[108] S. Barak, A. Arjmand, and S. Ortobelli, "Fusion of multiple diverse predictors in stock market," *Information Fusion*, vol. 36, pp. 90–102, 2017.

[109] B. Weng, L. Lu, X. Wang, F. M. Megahed, and W. Martinez, "Predicting short-term stock prices using ensemble methods and online data sources," *Expert Systems with Applications*, vol. 112, pp. 258 – 273, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417418303622

[110] Y. Chen and Y. Hao, "A feature weighted support vector machine and k-nearest neighbor algorithm for stock market indices prediction," *Expert Systems with Applications*, vol. 80, pp. 340–355, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417417301367

[111] J. L. Ticknor, "A bayesian regularized artificial neural network for stock market forecasting," *Expert Systems with Applications*, vol. 40(14), pp. 5501–5506, 2013.

[112] G. Tingwei and C. Yueting, "Improving stock closing price prediction

using recurrent neural network and technical indicators," *Neural Computation*, vol. 30(10), pp. 2833–2854, 2018.

[113] W. Long, Z. Lu, and L. Cui, "Deep learning-based feature engineering for stock price movement prediction," *Knowledge-Based Systems*, vol. 164, pp. 163–173, 2019.

[114] T. L. Sandritter, B. L. Jones, G. L. Kearns, and J. A. Lowry, *Nelson Textbook of Pediatrics*.

[115] M. Allison, "Reinventing clinical trials," *Nature biotechnology*, vol. 30(1), pp. 41–49, 2012.

[116] J. C. Bouvy, M. L. D. Bruin, and M. A. Koopmanschap, "Epidemiology of adverse drug reactions in europe: a review of recent observational studies," *Drug Safety*, vol. 38(5), pp. 437–453, 2015.

[117] J. D. H. Jonathan H. Watanabe, Terry McInnis, "Cost of prescription drug-related morbidity and mortality," *The Annals of pharmacotherapy*, vol. 52(9), pp. 829—-837, 2018.

[118] F. Zhang, B. Sun, X. Diao, W. Zhao, and T. Shu, "Prediction of adverse drug reactions based on knowledge graph embedding," *BMC Medical Informatics and Decision Making*, vol. 21, 2021.

[119] D. Sridhar, S. Fakhraei, and L. Getoor, "A probabilistic approach for collective similarity-based drug–drug interaction prediction," *Bioinformatics*, vol. 32, no. 20, pp. 3175–3182, 06 2016.

[120] M. Yu, S. Kim, Z. Wang, S. Hall, and L. Li, "A bayesian meta-analysis on published sample mean and variance pharmacokinetic data with application to drug–drug interaction prediction," *Journal of Biopharmaceutical Statistics*, vol. 18, no. 6, pp. 1063–1083, 2008.

[121] W. Zhang, K. Jing, F. Huang, Y. Chen, B. Li, J. Li, and J. Gong, "Sflln: A sparse feature learning ensemble method with linear neighborhood regularization for predicting drug–drug interactions," *Information Sciences*, vol. 497, pp. 189–201, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025519304116

[122] L. H. Dang, N. T. Dung, L. X. Quang, L. Q. Hung, N. H. Le, N. T. N. Le, N. T. Diem, N. T. T. Nga, S.-H. Hung, and N. Q. K. Le, "Machine learning-based prediction of drug-drug interactions for histamine antagonist using hybrid chemical features," *Cells*, vol. 10, no. 11, 2021. [Online]. Available: https://www.mdpi.com/2073-4409/10/11/3092

[123] R. Celebi, H. Uyar, E. Yasar, O. Gumus, O. Dikenelli, and M. Dumontier, "Evaluation of knowledge graph embedding approaches for drug-drug interaction prediction in realistic settings," *BMC Bioinformatics*, vol. 20, p. 726, 2019.

[124] H. Yu, W. Dong, and J. Shi, "Raneddi: Relation-aware network embedding for drug-drug interaction prediction," *Information Sciences*, vol. 582, pp. 167–180, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025521009294

[125] S. K. Sahu and A. Anand, "Drug-drug interaction extraction from biomedical texts using long short-term memory network," *Journal of Biomedical Informatics*, vol. 86, pp. 15–24, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1532046418301606

[126] S. Liu, Y. Zhang, Y. Cui, Y. Qiu, Y. Deng, Z. M. Zhang, and W. Zhang, "Enhancing drug-drug interaction prediction using deep attention neural networks," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, pp. 1–1, 2022.

[127] M. Rezaul Karim, M. Cochez, J. Jares, M. Uddin, O. Beyan, and S. Decker, "Drug-drug interaction prediction based on knowledge graph embeddings and convolutional-lstm network," in *ACM-BCB 2019 - Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*. Association for Computing Machinery, Inc, Sep. 2019, pp. 113–123.

[128] W. Zhang, Y. Chen, D. Li, and X. Yue, "Manifold regularized matrix factorization for drug-drug interaction prediction," *Journal of Biomedical Informatics*, vol. 88, pp. 90–97, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1532046418302144

[129] A. Mongia, S. Jain, É. Chouzenoux, and A. Majumdar, "Deepvir - graphical deep matrix factorization for "in silico" antiviral repositioning: Application to covid-19," *ArXiv*, vol. abs/2009.10333, 2020.

[130] J.-Y. Shi, H. Huang, J.-X. Li, P. Lei, Y.-N. Zhang, K. Dong, and S.-M.

Yiu, "Tmfuf: a triple matrix factorization-based unified framework for predicting comprehensive drug-drug interactions of new drugs," *BMC Bioinformatics*, vol. 19, p. 411, 2018.

[131] J.-Y. Shi, H. Huang, J.-X. Li, P. Lei, Y.-N. Zhang, and S.-M. Yiu, "Predicting comprehensive drug-drug interactions for new drugs via triple matrix factorization," in *Bioinformatics and Biomedical Engineering*, I. Rojas and F. Ortuño, Eds. Cham: Springer International Publishing, 2017, pp. 108–117.

[132] Y. Zhang, Y. Qiu, Y. Cui, S. Liu, and W. Zhang, "Predicting drug-drug interactions using multi-modal deep auto-encoders based network embedding and positive-unlabeled learning," *Methods*, vol. 179, pp. 37–46, 2020, interpretable machine learning in bioinformatics. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1046202319303421

[133] X. Lin, Z. Quan, Z.-J. Wang, T. Ma, and X. Zeng, "Kgnn: Knowledge graph neural network for drug-drug interaction prediction," in *IJCAI*, 2020.

[134] Y. Ding, J. Tang, and F. Guo, "Identification of drug-target interactions via multiple information integration," *Information Sciences*, vol. 418-419, pp. 546–560, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025517307776

[135] B. Tanoori, M. Z. Jahromi, and E. G. Mansoori, "Drug-target continuous binding affinity prediction using multiple sources of information," *Expert*

Systems with Applications, vol. 186, p. 115810, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417421011787

[136] T. Turki and Y. h. Taguchi, "Machine learning algorithms for predicting drugs–tissues relationships," *Expert Systems with Applications*, vol. 127, pp. 167–186, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417419301186

[137] P. G. Sun, Y. N. Quan, Q. G. Miao, and J. Chi, "Identifying influential genes in protein–protein interaction networks," *Information Sciences*, vol. 454-455, pp. 229–241, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025518303487

[138] B. Yu, C. Chen, X. Wang, Z. Yu, A. Ma, and B. Liu, "Prediction of protein–protein interactions based on elastic net and deep forest," *Expert Systems with Applications*, vol. 176, p. 114876, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417421003171

[139] H.-C. Lee, S.-W. Huang, and E. Y. Li, "Mining protein–protein interaction information on the internet," *Expert Systems with Applications*, vol. 30, no. 1, pp. 142–148, 2006, intelligent Bioinformatics Systems. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417405002496

[140] J. Zhang, C. Li, Y. Lin, Y. Shao, and S. Li, "Computational drug repositioning using collaborative filtering via multi-source fusion," *Expert*

Systems with Applications, vol. 84, pp. 281–289, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417417303202

[141] G. Chao, S. Sun, and J. Bi, "A survey on multiview clustering," *IEEE Transactions on Artificial Intelligence*, vol. 2, no. 2, pp. 146–168, 2021.

[142] S. Bickel and T. Scheffer, "Multi-view clustering," in *Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM 2004)*, 2004, pp. 19–26.

[143] X. Yi, Y. Xu, and C. Zhang, "Multi-view em algorithm for finite mixture models," in *Pattern Recognition and Data Mining*, S. Singh, M. Singh, C. Apte, and P. Perner, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 420–425.

[144] D. Lashkari and P. Golland, "Convex clustering with exemplar-based models," in *Advances in Neural Information Processing Systems*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds., vol. 20. Curran Associates, Inc., 2007.

[145] A. Kumar and H. D. III, "A co-training approach for multi-view spectral clustering," in *Proceedings of the 28th International Conference on International Conference on Machine Learning (ICML 2011)*. Madison, WI, USA: Omnipress, 2011, p. 393–400.

[146] J. Sun, J. Lu, T. Xu, and J. Bi, "Multi-view sparse co-clustering via proximal alternating linearized minimization," in *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, ser. Pro-

ceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37, Lille, France, 07–09 Jul 2015, pp. 757–766.

[147] T. Liu, "Guided co-training for large-scale multi-view spectral clustering," *CoRR*, vol. abs/1707.09866, 2017. [Online]. Available: http://arxiv.org/abs/1707.09866

[148] W. Cai, H. Zhou, and L. Xu, "A multi-view co-training clustering algorithm based on global and local structure preserving," *IEEE Access*, vol. 9, pp. 29 293–29 302, 2021.

[149] A. Kumar, P. Rai, and H. Daume, "Co-regularized multi-view spectral clustering," in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds., vol. 24. Curran Associates, Inc., 2011.

[150] Y. Ye, X. Liu, J. Yin, and E. Zhu, "Co-regularized kernel k-means for multi-view clustering," in *Proceedings of the 23rd International Conference on Pattern Recognition (ICPR 2016)*, 2016, pp. 1583–1588.

[151] M. Brbić and I. Kopriva, "Multi-view low-rank sparse subspace clustering," *Pattern Recognition*, vol. 73, pp. 247–258, 2018.

[152] C. Zhang, H. Fu, Q. Hu, X. Cao, Y. Xie, D. Tao, and D. Xu, "Generalized latent multi-view subspace clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 1, pp. 86–99, 2020.

[153] J. Tan, Y. Shi, Z. Yang, C. Wen, and L. Lin, "Unsupervised multi-view

clustering by squeezing hybrid knowledge from cross view and each view," *IEEE Transactions on Multimedia*, vol. 23, pp. 2943–2956, 2021.

[154] S. Yu, L. Tranchevent, X. Liu, W. Glanzel, J. A. Suykens, B. De Moor, and Y. Moreau, "Optimized data fusion for kernel k-means clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 5, pp. 1031–1039, 2012.

[155] X. Chen, X. Xu, J. Z. Huang, and Y. Ye, "Tw-k-means: Automated two-level variable weighting clustering algorithm for multiview data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 4, pp. 932–944, 2013.

[156] X. Cai, F. Nie, and H. Huang, "Multi-view k-means clustering on big data," in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI 2013)*. AAAI Press, 2013, p. 2598–2604.

[157] H. Liu and Y. Fu, "Consensus guided multi-view clustering," *ACM Trans. Knowl. Discov. Data*, vol. 12, no. 4, apr 2018. [Online]. Available: https://doi.org/10.1145/3182384

[158] T. Joachims, N. Cristianini, and J. Shawe-Taylor, "Composite kernels for hypertext categorisation," in *Proceedings of the 18th International Conference on Machine Learning (ICML 2001)*, 01 2001, pp. 250–257.

[159] T. Zhang, A. Popescul, and B. Dom, "Linear prediction models with graph regularization for web-page categorization," in *Proceedings of the*

12th ACM International Conference Knowledge Discovery Data Mining (SIGKDD 2006), 2006, p. 821–826.

[160] G. Chao and S. Sun, "Multi-kernel maximum entropy discrimination for multi-view learning," *Intelligence Data Analysis*, vol. 20, no. 3, p. 481–493, jan 2016.

[161] M. B. Blaschko and C. H. Lampert, "Correlational spectral clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, 2008, pp. 1–8.

[162] W. Shao, L. He, C.-t. Lu, and P. S. Yu, "Online multi-view clustering with incomplete views," in *Proceedings of the IEEE International Conference on Big Data (Big Data 2016)*, 2016, pp. 1012–1017.

[163] X. Yang, C. Deng, Z. Dang, and D. Tao, "Deep multiview collaborative clustering," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[164] J. Xu, Y. Ren, G. Li, L. Pan, C. Zhu, and Z. Xu, "Deep embedded multi-view clustering with collaborative training," *Information Sciences*, vol. 573, pp. 279–290, 2021.

[165] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[166] H. Zhang, G. Lu, M. Zhan, and B. Zhang, "Semi-supervised classification

of graph convolutional networks with laplacian rank constraints," *Neural Processing Letters*, vol. 54, pp. 1–12, 08 2022.

[167] S. Fan, X. Wang, C. Shi, E. Lu, K. Lin, and B. Wang, "One2multi graph autoencoder for multi-view graph clustering," in *In Proceedings of The Web Conference 2020 (WWW '20)*, 04 2020, pp. 3070–3076.

[168] A. Goel, A. Majumdar, E. Chouzenoux, and G. Chierchia, "Deep convolutional k-means clustering," in *Proceedings of the IEEE International Conference on Image Processing (ICIP 2022)*, Bordeaux, France, 2022, pp. 211–215.

[169] A. Goel and A. Majumdar, "Transformed k-means clustering," in *Proceedings of the 29th European Signal Processing Conference (EUSIPCO 2021)*, 2021, pp. 1526–1530.

[170] C. Bauckhage, "K-means clustering is matrix factorization," *arXiv preprint arXiv:1512.07548*, 2015.

[171] K. Zhan, C. Zhang, J. Guan, and J. Wang, "Graph learning for multiview clustering," *IEEE Transactions on Cybernetics*, vol. 48, no. 10, pp. 2887–2895, 2018.

[172] D. J. Trosten, S. Lokse, R. Jenssen, and M. Kampffmeyer, "Reconsidering representation alignment for multi-view clustering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2021)*, Los Alamitos, CA, USA, jun 2021, pp. 1255–1265.

[173] X. Peng, S. Xiao, J. Feng, W.-Y. Yau, and Z. Yi, "Deep subspace clustering with sparsity prior." in *IJCAI*, 2016, pp. 1925–1931.

[174] X. Peng, J. Feng, J. T. Zhou, Y. Lei, and S. Yan, "Deep subspace clustering," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 12, pp. 5509–5521, 2020.