End-to-End Trainable Deep Tensor Completion Algorithms with Multimodal Data Fusion for Anticancer Drug Synergy Prediction

by

Judah Stephen Zammit

A Thesis to the Faculty of Graduate Studies of

The University of Manitoba

In partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

Department of Computer Science

University of Manitoba

Winnipeg

Copyright © 2023 by Judah Stephen Zammit

Abstract

Computational algorithms for anticancer drug synergy prediction (ACDSP) have the potential to discover novel anticancer drug combinations. Most of the algorithms are based on deep learning (DL) and use chemical features of anticancer drugs and multiomic features of cancer cell lines. Deep tensor completion (DTC)-based ACDSP algorithms achieve similar performance to DL-based ACDSP algorithms without using chemical or multiomic features. However, current DTC-based ACDSP algorithms are trained in two separate steps, potentially limiting their performance. In this M.Sc. thesis, I develop a novel DTC-based ACDSP algorithm, named Deep Selective CANDECOMP/PARAFAC (DSCP), which can be trained in an end-to-end fashion. It was found that DSCP made significantly better performance on a synthetic tensor classification task, while using less resources than the non-end-to-end trainable version of DSCP. Furthermore, I also propose a new ACDSP algorithm that can utilize multimodal data fusion methods (e.g., intermediate fusion (IF)) to integrate tensor-based drug and cell line data with chemical and multiomic features, which is named SynergyIF. Finally, this thesis found evidence that many ACDSP algorithms do not utilize the chemical and biological structure of their input features, instead treating these features as unique identifiers. This was done by examining the effects of randomizing ACDSP algorithms' input data on their predictive performance. Overall, although the thesis demonstrates some advantages to integrating deep learning and tensor completion algorithms in an end-to-end fashion, more work is warranted to fuse multimodal data under the learning framework.

Acknowledgments

Dr. Pingzhao Hu has my endless gratitude for his tireless supervision. He gave me a chance to work on an academic project when I had no experience and I have learned nearly every academic skill I have from him. He has spent countless hours making me a better researcher, and I am deeply thankful for this. Dr. Carson Kai-Sang Leung always made himself available to me for advice and provided me with detailed and timely feedback that improved this thesis and many other projects immeasurably.

My collaborations with fellow students in Dr. Hu's lab have been invaluable to my writing of this thesis. I'd like to express my gratitude specifically to Qian Liu, who has been a fantastic mentor throughout both my undergraduate and M.Sc. degree. I'd like to extend my deep appreciation to the faculty members of the department of computer science for providing me with helpful guidance at many critical times in my academic life. In particular, I'd like to thank Robert Guderian, Dr. Cuneyt Akcora, Dr. Meng-Cheng Lau, Dr. Yang Wang, Dr. Pourang Irani, and Dr. Shahin Kamali. Additionally, I'd like to thank Dr. Olivier Tremblay-Savard and Dr. Yiming Qian for sitting on my advisory committee and providing me with illuminating feedback.

My family's support and encouragement has been invaluable to me during my M.Sc. and the times I spent playing board games with my friends, Myles Vasas and Zachary Vasas, gave me much needed rest from the pursuit of my M.Sc. Finally, my partner, friend, and support, Anamelechi Stanely stood by my side through the toughest times of these past two years and made every day warmer and happier. Without her companionship and love, I surely would not have been able to write this thesis.

Table of Contents

Abstract i

Acknowledgments ii

Table of Contents iii

List of Tables viii

List of Figures xiv

List of Abbreviations xvi

Chapter 1 Introduction 1

1.1 Background 1

1.2 Related Work 3

1.2.1 Tensor Decomposition 3

1.2.2 Tensor Completion 5

1.2.2.1 Linear Tensor Completion 5

1.2.2.2 Deep Tensor Completion 6

1.2.2.3 Tensor Completion with Auxiliary Data 9

1.2.3 In Silico Drug Combination Synergy Prediction 10

1.2.3.1 Traditional Algorithms 10

1.2.3.2 Tensor-Based Algorithms 11

1.2.4 Multimodal Data Fusion 12

1.3 Motivation and Research Objectives 12

1.4 Thesis Organization 13

Chapter 2 Anticancer Drug Synergy Prediction via End-to-End Deep Tensor Decomposition 14

2.1 Introduction 14

2.2 Materials and Methods 15

2.2.1 Notations 15

2.2.2 Tensor Decomposition Algorithms 16

2.2.2.1 CANDECOMP/PARAFAC Decomposition (CPD) 16

2.2.2.2 CP-WOPT 17

2.2.2.3 Deep Tensor Factorization 17

2.2.3 SCP: Selective CANDECOMP/PARAFAC 20

2.2.4 Deep Neural Network 22

2.2.5 Hyperparameter Description 25

2.3 Data Sources and Algorithm Evaluation Strategy 25

2.3.1 Data Sources 26

2.3.1.1 Drug Combination Compound Screens 26

2.3.1.2 Drug and Cell Line Auxiliary Features 27

2.3.1.3 Synthetic Data Generation 28

2.3.2 Cross-Validation Strategy 29

2.3.3 Compared Algorithms 30

2.3.4 Metrics 32

2.3.5 Hyperparameter Optimization 35

2.3.6 Test Set Inference 35

2.3.7 Software and Hardware 36

2.4 Results....36
2.4.1 Synthetic Data Analysis....36
2.4.2 Merick and Co Compound Screen....39
2.4.2.1 Leave-Drug-Combination-Out (LDCO)....39
2.4.2.2 Leave-Drug-Combination-Out (LDO)....42
2.4.2.3 Leave-Cell-Line-Out (LCLO)....46
2.4.3 NCI-ALMANAC Compound Screen....50
2.4.3.1 Leave-Drug-Combination-Out (LDCO)....50
2.5 Discussion....54
2.6 Summary....55
Chapter 3 Artificial Intelligence-Driven Multimodal Data Fusion Techniques for Deep Tensor Completion-Based Anticancer Drug Synergy Prediction....57
3.1 Introduction....57
3.2 Materials and Methods....58
3.2.1 Hyperparameter Description....62
3.3 Data Sources and Algorithm Evaluation Strategy....62
3.4 Results....64
3.4.1 Merick and Co Compound Screen....64
3.4.1.1 Leave-Drug-Combination-Out (LDCO)....64
3.4.1.2 Leave-Drug-Combination-Out (LDO)....69
3.4.1.3 Leave-Cell-Line-Out (LCLO)....74
3.4.2 NCI-ALMANAC Compound Screen....82
3.4.2.1 Leave-Drug-Combination-Out (LDCO)....82

3.5 Discussion 87

3.6 Summary 88

Chapter 4 Physicochemical, Molecular Graph, Multiomic and Tensor-Based Features Offer Little Increase in Predictive Performance over Randomized Features for Anticancer Drug Synergy Prediction 90

4.1 Introduction 90

4.2 Materials and Methods 91

4.3 Data Sources and Algorithm Evaluation Strategy 93

4.3.1 Drug Combination Compound Screens- 94

4.3.2 Randomization Evaluation Strategy 95

4.4 Results 96

4.4.1 Merick and Co Compound Screen 96

4.4.1.1 Leave-Drug-Combination-Out (LDCO) 96

4.4.1.2 Leave-Drug-Out (LDO) 98

4.4.1.3 Leave-Cell-Line-Out (LCLO) 100

4.4.2 NCI-ALMANAC Compound Screen 101

4.4.2.1 Leave-Drug-Combination-Out (LDCO) 101

4.4.3 DrugComb Compound Screen 103

4.4.3.1 Leave-Drug-Combination-Out (LDCO) 103

4.4.4 Unique Identifier Experiment 104

4.5 Discussion 106

4.6 Summary 109

Chapter 5 Conclusion and Future Work 111

5.1 Conclusion 111

5.2 Future Work 112

Bibliography 114

Supplementary 125

List of Tables

Table 2.1 A summary of the algorithms, data sources and cross-validation strategies considered in this chapter. 26

Table 2.2 The results of the hyperparameter search for DSCP on the Merick and Co compound screen using the LDCO cross-validation strategy. 39

Table 2.3 The classification results of our compared algorithms on the Merick and Co compound screen using LDCO cross-validation. 41

Table 2.4 The regression results of our compared algorithms on the Merick and Co compound screen using LDCO cross-validation. 42

Table 2.5 The results of the hyperparameter search for DSCP on the Merick and Co compound screen using the LDO cross-validation strategy. 43

Table 2.6 The classification results of our compared algorithms on the Merick and Co compound screen using LDO cross-validation. 45

Table 2.7 The regression results of our compared algorithms on the Merick and Co compound screen using LDO cross-validation. 46

Table 2.8 The results of the hyperparameter search for DSCP on the Merick and Co compound screen using the LCLO cross-validation strategy. 47

Table 2.9 The classification results of our compared algorithms on the Merick and Co compound screen using LCLO cross-validation. 49

Table 2.10 The regression results of our compared algorithms on the Merick and Co compound screen using LCLO cross-validation. 50

Table 2.11 The results of the hyperparameter search for DSCP on the NCI-ALMANAC compound screen using the LDCO cross-validation strategy. 51

Table 2.12 The classification results of our compared algorithms on the NCI-ALMANAC compound screen using LDCO cross-validation. 53

Table 2.13 The regression results of our compared algorithms on the NCI-ALMANAC compound screen using LDCO cross-validation. 54

Table 3.1 A summary of the algorithms, data sources and cross-validation strategies considered in this chapter. 63

Table 3.2 The results of the hyperparameter search for SynergyIF on the Merick and Co compound screen using the LDCO cross-validation strategy. 65

Table 3.3 The classification results of our compared algorithms on the Merick and Co compound screen using LDCO cross-validation. 68

Table 3.4 The regression results of our compared algorithms on the Merick and Co compound screen using LDCO cross-validation. 69

Table 3.5 The results of the hyperparameter search for SynergyIF on the Merick and Co compound screen using the LDO cross-validation strategy. 70

Table 3.6 The classification results of our compared algorithms on the Merick and Co compound screen using LDO cross-validation. 73

Table 3.7 The regression results of our compared algorithms on the Merick and Co compound screen using LDO cross-validation. 74

Table 3.8 The results of the hyperparameter search for SynergyIF on the Merick and Co compound screen using the LCLO cross-validation strategy. 75

Table 3.9 The classification results of our compared algorithms on the Merick and Co compound screen using LCLO cross-validation. 81

Table 3.10 The regression results of our compared algorithms on the Merick and Co compound screen using LCLO cross-validation. 82

Table 3.11 The results of the hyperparameter search for SynergyIF on the NCI-ALMANAC compound screen using the LDCO cross-validation strategy. 83

Table 3.12 The classification results of our compared algorithms on the NCI-ALMANAC compound screen using LDCO cross-validation. 86

Table 3.13 The regression results of our compared algorithms on the NCI-ALMANAC compound screen using LDCO cross-validation. 87

Table 4.1 A summary of the algorithms, data sources, cross-validation strategies, metrics, and software considered in this chapter. 94

Table 4.2 A comparison of the classification performance of ACDSP algorithms with and without randomized input data on the Merick and Co compound screen using the LDCO cross-validation strategy. 98

Table 4.3 A comparison of the regression performance ACDSP algorithms with and without randomized input data on the Merick and Co compound screen using the LDCO cross-validation strategy. 98

Table 4.4 A comparison of the classification performance of ACDSP algorithms with and without randomized input data on the Merick and Co compound screen using the LDO cross-validation strategy. 99

Table 4.5 A comparison of the regression performance ACDSP algorithms with and without randomized input data on the Merick and Co compound screen using the LDO cross-validation strategy. 100

Table 4.6 A comparison of the classification performance of ACDSP algorithms with and without randomized input data on the Merick and Co compound screen using the LCLO cross-validation strategy. 101

Table 4.7 A comparison of the regression performance of ACDSP algorithms with and without randomized input data on the Merick and Co compound screen using the LCLO cross-validation strategy. 101

Table 4.8 A comparison of the classification performance of ACDSP algorithms with and without randomized input data on the NCI-ALMANAC compound screen using the LDCO cross-validation strategy. 102

Table 4.9 A comparison of the regression performance ACDSP algorithms with and without randomized input data on the NCI-ALMANAC compound screen using the LDCO cross-validation strategy. 103

Table 4.10 A comparison of the regression performance of ACDSP algorithms with and without randomized input data on the DrugComb compound screen using the LDCO cross-validation strategy. 104

Table 4.11 A comparison of the regression performance of ACDSP algorithms with and without randomized input data on the DrugComb compound screen using the LDCO cross-validation strategy. 104

Table 4.12 A comparison of the regression performance of DeepSynergy with and without identity matrix input data, on every dataset and cross-validation strategy. 105

Table 4.13 A comparison of the classification performance of DeepSynergy with and without identity matrix input data, on every dataset and cross-validation strategy. 106

Table S.1 The results of the hyperparameter search for CP-Wopt on the Merick and Co compound screen using the LDCO cross-validation strategy. 126

Table S.2 The results of the hyperparameter search for DTF on the Merick and Co compound screen using the LDCO cross-validation strategy. 126

Table S.3 The results of the hyperparameter search for DeepSynergy on the Merick and Co compound screen using the LDCO cross-validation strategy. 127

Table S.4 The results of the hyperparameter search for CoSTCo on the Merick and Co compound screen using the LDCO cross-validation strategy. 128

Table S.5 The results of the hyperparameter search for PRODeepSyn on the Merick and Co compound screen using the LDCO cross-validation strategy. 128

Table S.6 The results of the hyperparameter search for CP-Wopt on the Merick and Co compound screen using the LDO cross-validation strategy. 129

Table S.7 The results of the hyperparameter search for DTF on the Merick and Co compound screen using the LDO cross-validation strategy. 129

Table S.8 The results of the hyperparameter search for DeepSynergy on the Merick and Co compound screen using the LDO cross-validation strategy. 130

Table S.9 The results of the hyperparameter search for CoSTCo on the Merick and Co compound screen using the LDO cross-validation strategy. 131

Table S.10 The results of the hyperparameter search for PRODeepSyn on the Merick and Co compound screen using the LDO cross-validation strategy. 131

Table S.11 The results of the hyperparameter search for CP-Wopt on the Merick and Co compound screen using the LCLO cross-validation strategy. 132
Table S.12 The results of the hyperparameter search for DTF on the Merick and Co compound screen using the LCLO cross-validation strategy. 132
Table S.13 The results of the hyperparameter search for DeepSynergy on the Merick and Co compound screen using the LCLO cross-validation strategy. 133
Table S.14 The results of the hyperparameter search for CoSTCo on the Merick and Co compound screen using the LCLO cross-validation strategy. 134
Table S.15 The results of the hyperparameter search for PRODeepSyn on the Merick and Co compound screen using the LCLO cross-validation strategy. 134
Table S.16 The results of the hyperparameter search for CP-Wopt on the NCI-ALMANAC compound screen using the LDCO cross-validation strategy. 135
Table S.17 The results of the hyperparameter search for DTF on the NCI-ALMANAC compound screen using the LDCO cross-validation strategy. 135
Table S.18 The results of the hyperparameter search for DeepSynergy on the NCI-ALMANAC compound screen using the LDCO cross-validation strategy. 136
Table S.19 The results of the hyperparameter search for CoSTCo on the NCI-ALMANAC compound screen using the LDCO cross-validation strategy. 136
Table S.20 The results of the hyperparameter search for PRODeepSyn on the NCI-ALMANAC compound screen using the LDCO cross-validation strategy. 137

List of Figures

Figure 1.1 An illustration showing how drug synergy data can be represented as an order-3 tensor. 2

Figure 2.1 The workflow of the previous DTF algorithm. (A) Data collection and preprocessing. 19

Figure 2.2 An illustration of SCP. 21

Figure 2.3 The workflow of the proposed DSCP algorithm. 23

Figure 2.4 A depiction of the various cross-validation strategies that can be employed for ACDSP. 30

Figure 2.5 The ROC-AUC of the DTF and DSCP algorithms, with varying rank, on the synthetic tensor task. 37

Figure 2.6 The ROC-AUC of the DTF and DSCP algorithms on the synthetic tensor task with varying levels of missing data. 38

Figure 2.7 The PR-AUC of the DTF and DSCP algorithms, with varying class imbalance, on the synthetic tensor task. 38

Figure 2.8 A summary of the classification and regression results on the Merick and Co compound screen using LDCO cross-validation. 40

Figure 2.9 A summary of the classification and regression results on the Merick and Co compound screen using LDO cross-validation. 44

Figure 2.10 A summary of the classification and regression results on the Merick and Co compound screen using LCLO cross-validation. 48

Figure 2.11 A summary of the classification and regression results on the NCI-ALMANAC compound screen using LDCO cross-validation. 52

Figure 3.1 A legend for SynergyIF 60

Figure 3.2 A depiction of SynergyIF. 61

Figure 3.3 A summary of the classification and regression results on the Merick and Co compound screen using LDCO cross-validation. 67

Figure 3.4 A summary of the classification and regression results on the Merick and Co compound screen using LDO cross-validation. 72

Figure 3.5 A scatter plot of SynergyIF's predictions against the ground truth. 77

Figure 3.6 A scatter plot of SynergyIF's predictions against the average for each drug pair across every cell line in the training set. 78

Figure 3.7 A scatter plot of SynergyIF's predictions against the ground truth with the average subtracted from both. 79

Figure 3.8 A summary of the classification and regression results on the Merick and Co compound screen using LCLO cross-validation. 80

Figure 3.9 A summary of the classification and regression results on the NCI-ALMANAC compound screen using LDCO cross-validation. 84

Figure 4.1 A depiction of the randomization strategy used in this chapter. 92

Figure 4.2 A depiction of the permutation strategy used in this chapter. 93

List of Abbreviations

Abbreviations	Definitions
ACDSP	anticancer drug synergy prediction
AUC	Area Under Curve
BACC	Balanced Accuracy
CANDECOMP	Canonical Decomposition
CMTF	Coupled Matrix/Tensor Factorization
CoSTCo	Convolutional Sparse Tensor Completion
CP	CANDECOMP/PARAFAC
CPD	CANDECOMP/PARAFAC Decomposition
DDS	Drug-Drug Synergy
DL	Deep Learning
DNN	Deep Neural Network
DR	Dropout Rate
DSCP	Deep Selective CANDECOMP/PARAFAC
DTC	Deep Tensor Completion
DTF	Deep Tensor Factorization
GCN	Graph Convolutional Network
GPU	Graphical Processing Unit
IF	Intermediate Fusion
LCLO	Leave Cell Line Out
LDCO	Leave Drug Combination Out

LDO	Leave Drug Out
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MDAE	Median Absolute Error
MLCTR	Multi-Layer Matrix Factorization Neural Networks for Coupled Tensor Reconstruction
MLP	Multi-Layer Perceptron
NCI	National Cancer Institute
NTDMDA	Neural Tensor Decomposition-Based Method for the Prediction of MiRNA-Disease Associations
NTF	Neural Tensor Factorization
PARAFAC	Parallel Factor Analysis
PPI	Protien-Protien Interaction
PR	Precision-Recall
RMSE	Root Mean Squared Error
ROC	Characteristic Receiving Curve
SCP	Selective CANDECOMP/PARAFAC
SOTA	State-of -the-Art
TC	Tensor Completion
TD	Tensor Decomposition
WOPT	Weighted Optimization

Chapter 1 Introduction

1.1 Background

Combination therapy has been shown to be highly effective at treating cancer [1], [2]. Two drugs are considered to be synergistic when administering them together results in a greater reduction of cancerous tissue then would be expected by adding the reduction of cancerous tissue resulting from applying each drug individually. In essence, a drug pair is synergistic when it is greater than the sum of its parts. The immense number of possible drug combinations and the cost of *in vivo* and *in vitro* experimentation, makes the discovery of new synergistic drug combinations tremendously difficult.

In silico approaches have grown in popularity as an answer to this problem. While wet lab experiments are expensive and slow, computers are cheap and fast. Among *in silico* approaches, algorithms based on DL tend to perform best [3]. It is necessary for these algorithms to gain a deep understanding of the drugs, their interactions with each other, and the target cell lines. In many scenarios, such an understanding is necessary to make an accurate prediction, however an appealing alternative exists. It is possible to infer the synergies between drugs that have not been tested together from previous experiments that they have been involved in. This type of reasoning requires no understanding of the drugs, or the cell line, involved and yet can generate predictions with accuracy that is comparable to DL-based ACDSP algorithms [3], [4].

A class of algorithms that can perform this style of reasoning is tensor completion (TC) [5]. The results of drug synergy experiments can be seen as a cube-like mathematical object known as an order-three tensor (**Figure 1.1**). Two axes of this tensor correspond to the drug and

the third axis corresponds to the cell line [4], [6]. Because not every drug combination has been tested, there will be a considerable number of missing values in this tensor. TC is the task of imputing the missing values of a tensor and is clearly applicable in this case. TC algorithms are numerous and have been applied with remarkable success, both inside and outside bioinformatics [7]–[9]. Despite the success of TC, there are only two algorithms that have been applied to ACDSP [4], [6].

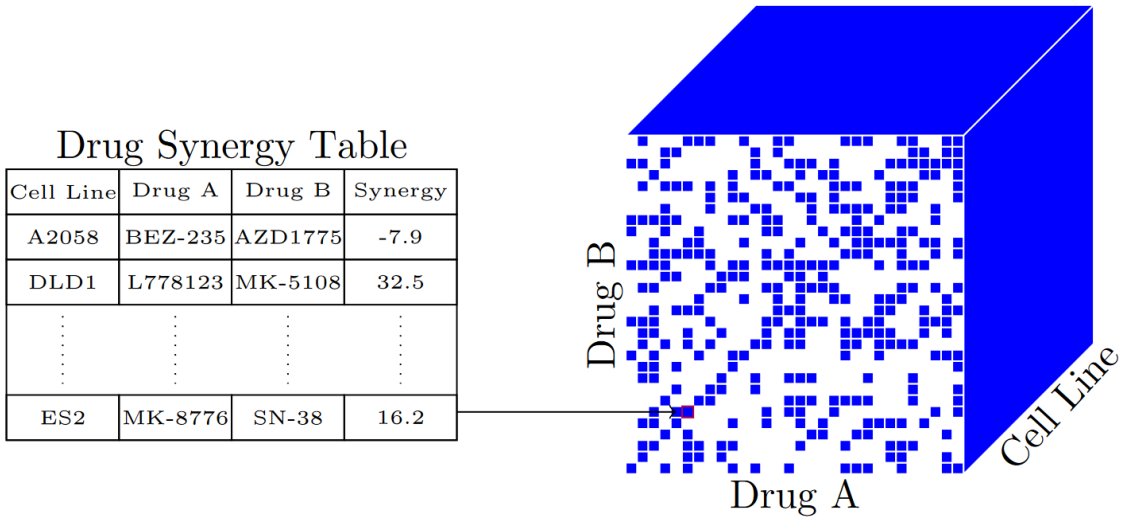

Figure 1.1 An illustration showing how drug synergy data can be represented as an order-3 tensor*. Blue entries indicate that the drug combination is observed, and white entries indicate that it is unobserved.*

These algorithms have several flaws. Because TC is typically based on a linear process, it is often too simple to capture the complex patterns present in anticancer drug combination data. Thus, there is immense potential in combining DL with TC. However, current DTC algorithms [4] use only the results of previous drug combination experiments to make their predictions. This implies that if a new drug, or cell line, is introduced to the dataset and no drug combination data is available, they will be unable to make any prediction [3]. This poses a significant drawback to

their practical applicability. While it is impractical to assume that all drugs will have drug combination data available, it is quite reasonable to assume that auxiliary data, such as physiochemical properties, are available. Additionally, current DTC algorithms are not trainable in an end-to-end manner. Thus, a DTC algorithm that, in addition to drug combination data, utilizes auxiliary data and is trainable end-to-end would greatly improve upon the practical utility of DTC algorithms for anti-cancer drug combination discovery. It is these flaws, a lack of auxiliary data and not being trainable in an end-to-end manner, that this thesis seeks to remedy.

This thesis found evidence that many ACDSP algorithms do not utilize the chemical and biological structure of their input features, instead treating these features as unique identifiers. It was found that DSCP made higher quality predictions on a synthetic tensor classification task, while using less resources than the non-end-to-end trainable version of DSCP. On the ACDSP task, we found that DSCP did not greatly improve upon the non-end-to-end trainable version and postulated that this was due to the fact that DSCP offers no advantage at generating unique identifiers for the drugs and cell lines. It was found that SynergyIF did not clearly improve upon ACDSP algorithms that utilize only one data source and postulated that this is due to none of the data sources being clearly better than randomized features.

1.2 Related Work

1.2.1 Tensor Decomposition

Tensor decomposition (TD) [10] is a crucial analysis tool for tensors. TD decomposes a tensor into a set of factors, effectively separating its various sources of variation. Where a matrix is a list of vectors, an order-3 tensor is a list of matrices. Furthermore, an order-N+1 tensor is a

list of order-N tensor. Each of the N axes of an order-N tensor corresponds to some entity. For example, the axes of an order-3 tensor may correspond to customers, items, and websites, and the tensor's entries may contain customer reviews. TD decomposes a tensor into a set of factors. This effectively separates the various sources of variation within a tensor.

Researchers have developed an immense number of TD algorithms, however we will focus only on the algorithms that are most used in TC, as that is our focus. CANDECOMP/PARAFAC decomposition (CPD) is arguably the most popular TD algorithm [11]. CPD works by finding a factor vector for each entity in the tensor, such that the dot product between the vectors will reconstruct the appropriate entry in the tensor. For an order-N tensor, these factor vectors are arranged into N factor matrices.

The Tucker decomposition [12] is a close second to CPD in terms of popularity. Like CPD, it will attempt to find factor vectors, but instead of reconstructing the entries with the dot product, it will take the tensor product between the factor matrices and a small core tensor. The result is two-fold. One, unlike CPD, it allows the number of columns for each factor matrix to be of varied sizes. Two, the tensor product is more expressive than the dot-product, allowing for the Tucker decomposition to represent complicated tensors in a more compressed format than CPD. However, CPD is often preferred over Tucker because the results of CPD are more easily interpreted.

Both CPD and Tucker decomposition assume that the entries of the tensor can be reconstructed through a linear process. For complex, real-world tensors, this linearity severely limits their abilities to decompose the tensors into information dense latent factors. Despite their drawbacks, these algorithms remain crucial to any discussion of tensor analysis, as they form the basis for nearly all state-of-the-art (SOTA) tensor analysis techniques.

1.2.2 Tensor Completion

Either due to noise or incomplete data collection, many tensors have a substantial number of entries whose values are unknown. The goal of TC is to impute these missing entries using the observed entries. While there are several approaches to TC, we will focus exclusively on TD-based Missing-Skipping TC. According to a review by Song et al. [5], The MS approach works by ignoring the missing values when performing TD. We focus on this approach because our primary goal is to develop DL-based TC, and MS-based TC is easily optimized with stochastic gradient descent. This makes integration with modern DL software packages effortless.

1.2.2.1 Linear Tensor Completion

A simple approach to TC is to perform TD on a partially observed tensor while ignoring the missing values. This is done by treating TD as an unconstrained optimization problem, weighted such that the missing values do not contribute to the loss function. For accurate imputation, we also must prevent the decomposition algorithm from overfitting to the observed data so that it will be forced to learn general patterns in the tensor. This can be done limiting the length of the factors, also known as applying the low-rank assumption, or applying regularization.

Using this technique, researchers have converted the most popular TD algorithms into TC algorithms. CP Weighted OPTimization (CP-WOPT) [13], Tucker-WOPT [14] and TT-WOPT [15] apply this technique to CPD, Tucker and tensor-train decomposition [16], respectively. Hong et al. [17] modify CP-WOPT for tensors that contain specific data types, such as the positive integers or binary values. They created GeneralCP, an excellent framework for coping with a vast array of data types.

The primary mechanism that allows TC algorithms to impute missing values accurately is their ability to extract patterns from the observed values that generalize to the missing values. Because these algorithms are based on linear TD, they are restricted to finding linear patterns. This is a tremendous limitation because the patterns present in real-world data are often not linear.

1.2.2.2 Deep Tensor Completion

Over the past several years, researchers have begun to develop TC algorithms based on DL. This is to solve the linear limitation of traditional TC algorithms. To the best of our knowledge, the research discussed in this section represents a complete list of (DTC) algorithms.

Linear TD reconstructs the original tensor by applying a linear function to the latent factors. An obvious DTC approach is to replace this linear function with some form of deep neural network. Liu et al. [18] note that sufficiently deep neural networks can reconstruct the observed values in the tensor using a completely *random* set of latent factors. Clearly, this approach to DTC is not guaranteed to find meaningful latent factors. Instead, they proposed Convolutional Sparse Tensor Completion (CoSTCo), a DTC algorithm that uses a shallow neural network, based on convolutional neural networks [19], that has theoretical guarantees to prevent overfitting. CoSTCo is remarkably easy to implement, computationally efficient and far more effective than linear TC. Replacing its convolutional layers with the incredibly successful attention mechanism [20], could further improve CoSTCo. Because CoSTCo's shallow structure may limit the complexity of the patterns it is able to capture, an algorithm that avoids overfitting without limiting the depth of the network may outperform CoSTCo.

Fang et al. [21] propose a DTC algorithm, called Streaming Probabilistic Deep tEnsoR factorization (SPIDER), that avoids over fitting by using a DNN with extensive regularization. Specifically, they use a Bayesian neural network with a spike and slab prior [22]. Though their

method is designed for streaming data, it is conceivable to use their approach for non-streaming data. Despite promising performance, the implementation of SPIDER is extremely complex, severely limiting its practical applicability. Additionally, such heavy regularization may limit the complexity of the patterns it is able to capture.

Both SPIDER and CoSTCo treat the latent factors as a learnable parameter and introduces non-linearity in the reconstruction phase of TC. An alternative approach is to introduce non-linearity to the latent factors themselves. This approach is called deep factor-based TC. Neural network-based Tensor Factorization (NTF) [23] uses this approach for tensors with one temporal axis. Even though any given time point will depend on the time points that came before it, CPD assumes that the latent factors for the temporal mode's entities are independent of each other. NTF remedies this by applying the Long Short-Term Memory (LSTM) [24] DL algorithm to the temporal latent factors before reconstructing the original tensor. This approach could be repeated for spatial tensors using convolutional layers instead of the LSTM. Additionally, replacing the LSTM with the more popular transformer [20] could significantly improve NTF's performance.

Uddin et al. [25] proposed Multi-Layer Matrix Factorization Neural Networks for Coupled Tensor Reconstruction (MLCTR), a deep factor-based TC algorithm that is intended for general tensors, not just temporal ones. They elect to use the CPD, but first they apply a deep non-linear process based off the ResNet [26] and matrix factorization. As well as considering the standard linear dot product for reconstruction, they also consider a multi-layer perceptron (MLP) [19]. The latter achieved superior results.

Instead of directly learning the latent factors, they can also be inferred from the tensor itself. TC algorithms that use this approach can be seen as a type of autoencoder [19] where the

inference phase is the encoder, and the reconstruction phase is the decoder. NeuralCP [27] uses the variational autoencoder (VAE) framework [28]. It first takes a specific entry from the tensor and uses it as input into an MLP. This MLP will output a distribution for the latent factors. A sample from this distribution is then used as input into an encoder MLP. NeuralCP tends to have lack luster performance when compared to CoSTCo, which may be explained by the fact that no effort is made to regularize the decoder network. Additionally, shifting the input by only one axis will change every latent factor. Ideally, it should only change the one latent factor corresponding to the axis that has change.

Variational AutoEncoder Bayesian Poisson-Gamma Tensor Factorization (VAE-BPTF) [29] is another DTC algorithm based on the VAE framework. By modelling the tensor's entries with the Poisson distribution, VAE-BPTF is specifically designed for tensors defined on the non-negative integers. Unlike NeuralCP, VAE-BPTF uses a mixture of encoder MLPs capable of ignoring missing values in the input tensor. However, VAE-BPTF is using a linear decoder, which may limit its performance.

Deep Tensor Factorization (DTF) [4] deviates from the DTC algorithms we have seen thus far. It first applies CP-WOPT to an order-three tensor. Because they do not use the low-rank assumption, their application of CP-WOPT only decomposes the tensor. It is not capable of imputing the missing values. To perform imputation, they train an MLP to predict the tensor's entry given its three associated factor vectors. The fact that these two steps are not trained in an end-to-end fashion may decrease the performance of DTF.

Though not strictly a DTC algorithm, Data Augmentation with INfluence function (DAIN) [30] is a sophisticated data augmentation approach for DTC algorithms. DAIN begins by training a DTC algorithm. It then uses the TracIn [31] influence function to calculate the

contribution that each latent factor makes to the imputation accuracy of tensor entries in a validation set. The entries associated with the most important entities are imputed and added to the tensor. A second DTC method is trained on this partially imputed tensor, and this process repeats for the desired number of iterations. This approach is reminiscent of student-teacher networks [32], a self-supervised learning algorithm. A formal exploration of this similarity may result in insights that increase DAIN's performance.

1.2.2.3 Tensor Completion with Auxiliary Data

TC can make accurate inferences about the missing values in a tensor by utilizing observed values that are related to the missing values. It is quite common, particularly in bioinformatics [3], for many entities in a tensor to have little or no related observed values. In this situation, it is quite difficult to make accurate inferences, even with DTC algorithms. Auxiliary data is used to overcome this issue. A tensor contains a set of entities for each of its axes. Auxiliary data provides additional information about those entities. This can be a matrix representing similarity between entities, additional features of the entities, or relational data between different sets of entities. It is also possible for auxiliary data to come in the form of a tensor.

TC algorithms that utilize auxiliary data tend to come in two broad categories: similarity regularization and coupled tensor-based [5]. Narita et al. [33] proposed the first similarity regularization algorithm. They use two types of regularization based on the similarity matrices. The first enforces that pairs of factor vectors corresponding to entities that are similar should be close together in Euclidean space. The second enforces that triplets of factor vectors corresponding to entities that are similar should be close together in Euclidean space. Acar et al. [34] proposed the first coupled tensor algorithm, Coupled Matrix/Tensor Factorization (CMTF).

This approach simultaneously applies TD to a set of tensors, or matrices, with overlapping sets of entities. Every tensor share factor matrices corresponding to these overlapping entities.

While most TC algorithms that utilize auxiliary data are linear, we will now discuss, to the best of our knowledge, the only two DTC algorithms. First, Neural Tensor Decomposition-based method for the prediction of MiRNA-Disease Associations (NTDMDA) [9] fuses auxiliary data into a tensor using a Gaussian kernel. It then simply applies the DTC algorithm CoSTCo [18]. The already discussed MLCTR algorithm [25] uses coupled tensors to utilize an order-three auxiliary tensor. There is a stark lack of DTC algorithms capable of utilizing auxiliary data. For example, there is none based on similarity regularization. This is a highly promising research direction, which will be pursued in this thesis.

1.2.3 In Silico Drug Combination Synergy Prediction

1.2.3.1 Traditional Algorithms

DL-based approaches have emerged as the preferred approach for ACDSP [3]. DeepSynergy [35] is a good representative of this type of approach. It predicts synergies between drugs with an MLP trained using a variety of multiomics data related to the drugs and cell lines. Recent advances include reducing the dimension of the multiomics data using an autoencoder before applying an MLP [36].

Algorithms that come after DeepSynergy tend to preserve the MLP to make the final prediction, choosing instead to extract drug and cell line features in more sophisticated manners. PRODeepSyn [37] makes uses of protein-protein interaction networks [38] and graph neural networks [39] to extract features for cell lines. HyperGraphSynergy [40] represents past known drug combination synergies as hyperedges in a hypergraph with drugs and cell lines as nodes.

HyperGraphSynergy then uses a hyper graph neural network [41], [42] to generate features for the drugs and cell lines.

1.2.3.2 Tensor-Based Algorithms

The algorithms in the previous section must gain a deep understanding of the drugs, their interaction with each other and the cancer types. By representing the synergies between drugs for various cancer types as a partially observed, order-three tensor with drugs along two axes and the target cell line along the third, we can infer the unknown drug synergies using TC. TC can make prediction of similar accuracy as DL-based algorithms while requiring no information about the drugs and cancer types [3].

Despite the clear advantage of TC for drug combination synergy prediction, only two algorithms exist for it. We have already discussed DTF, a DTC algorithm designed for drug synergy prediction. DTF maintains the MLP for the final prediction but forms the past anticancer drug synergy information into a tensor and uses TC to extract features for drugs and cell lines. The second algorithm is DrugCom [6], a linear TC algorithm that utilizes auxiliary data. Specifically, they utilize similarity matrices to form a coupled tensor and apply TC. Though DTF is highly non-linear, it does not use auxiliary or monotherapy data, and is not trainable in an end-to-end manner. Conversely, DrugCom is severely limited by the fact that it is linear but does effectively utilize auxiliary data. Though not TC algorithms, ComboFM [43] and ComboLTM [44] consider arranging drug synergies into an order-5 tensor with a raw dose-response matrix for every drug combination rather than just a synergy score. After creating this tensor, they apply factorization machines to predict the missing values in the tensor.

1.2.4 Multimodal Data Fusion

Many applications require the use of multiple data sources, thus requiring algorithms to fuse these different data modes together. Much biomedical literature of this type simply concatenates these data sources at the beginning of their algorithms. This style of multimodal data fusion is called Early Fusion. There is evidence [45]these Early Fusion-based strategies are a poor choice for biomedical applications. Intermediate Fusion (IF) and Late Fusion have been shown to be a better choice in many applications by allowing the algorithms to first discard information in the original data sources that are not useful for the task at hand. IF first embeds the data sources down to a low dimensional representation and then combines them. Late Fusion uses each data source to make different predictions and then combines those predictions.

1.3 Motivation and Research Objectives

DTC algorithms can be used to make highly accurate predictions of synergies between anticancer drugs. It cannot be used to make any predictions for a drug with no prior synergy data present. This greatly inhibits the practical utility of DTC algorithms for ACDSP. In this thesis, we aim to develop DTC algorithms capable of making predictions for drugs with no synergy data by accomplishing the following three aims:

Aim 1: Develop a DTC algorithm that is trainable in an end-to-end fashion.

Aim 2: Develop a DTC algorithm that effectively utilizes the auxiliary data of the drugs and cell lines.

Aim 3: Investigate how much utility physicochemical, molecular graph, multiomic and tensor-based features add to ACDSP algorithms.

1.4 Thesis Organization

Chapter 1 of this thesis reviews current ACDSP algorithms, current DTC algorithms and surrounding topics. This background information is used to propose a set of three research objectives in chapter 2. Chapter 3 introduces three anticancer drug synergy compound screens, three evaluation strategies and the ACDSP algorithms that we analyzed. In chapter 4 a novel DTC algorithm, named Deep Selective CANDECOMP/PARAFAC (DSCP), that is trainable in an end-to-end fashion is proposed. In addition to the ACDSP tasks laid out in chapter 3, DSCP is compared to a non-end-to-end trainable DTC algorithm on a synthetic dataset. Chapter 5 proposes an ACDSP algorithm, named SynergyIF, that fuses together tensor-based drug and cell line data with chemical and multiomic features, utilizing the multimodal data fusion technique, Intermediate Fusion (IF). Finally, chapter 6 performs an extensive analysis on how well ACDSP algorithms utilize the chemical and biological structure of their input data. It does this by randomizing each algorithm's input data and assessing the effect this had on the predictive performance of the algorithm.

Chapter 2 Anticancer Drug Synergy Prediction via End-to-End Deep Tensor Decomposition

2.1 Introduction

DTF has been shown to achieve exceptional performance on the ACDSP task. However, DTF is not trainable in an end-to-end manner. That is, the DL component and the TC component are trained in two separate steps. For this reason, DTF has no ability to selectively compress the most important information present in the drug synergy tensor. DTF instead resorts to compressing all the information in the tensor using 1000-dimensional latent factors. This ensures that no important information is missed but also that much irrelevant information is included. It has been repeatedly shown that the performance of DL algorithms often declines as the dimension of feature space increases, particularly if no new important information is being added [46]. This is known as the Curse-of-Dimensionality and is the reason that DL researchers and engineers seek to only use the most relevant and important features for their algorithms. Being able to learn to compress patterns present in the tensor that are the most important to ACDSP and ignore everything else has the potential to greatly reduce the dimensionality of the latent factors, mitigating the Curse-of-Dimensionality.

This could be achieved by training the TC and DL component of a DTC algorithm in an end-to-end fashion. However, the TC algorithm used by DTF, CP-WOPT, presents some problems for end-to-end training. In particular, it has no trainable parameters. A new TC algorithm capable of selectively compressing information in the tensor, based on a set of parameters, is needed to allow for end-to-end training.

We hypothesize that a DTC algorithm that is trainable in an end-to-end fashion will allow for superior classification of synergistic drug pairs by enabling the TC component to focus only on patterns in the drug synergy tensor that are relevant for ACDSP. To this end, we propose a novel TC algorithm called Selective CANDECOMP/PARAFAC (SCP) which introduces a set of parameter matrices. Doing so allows SCP to be trained in an end-to-end fashion, guided by the performance of the DL component. We name this algorithm Deep Selective CANDECOMP/PARAFAC (DSCP).

We compare the performance of DSCP to many other SOTA ACDSP algorithms [4], [13], [18], [35], [37], [40] on two high-quality unbiased compound screens [2], [47]. Finally, we perform an ablation study on a synthetic dataset, showing that end-to-end training is crucial for DTC.

2.2 Materials and Methods

2.2.1 Notations

Regarding tensors and their related concepts, we follow the notation set forward by Kolda and Bader [10] and direct the reader to their review of TD for complete definitions of the notations set forth in this section. A tensor generalizes matrices to higher orders. Specifically, a vector is an order-1 tensor, and a matrix is an order-2 tensor. In the same way that a matrix can be seen as a list of vectors, an order-3 tensor can be seen as a list of matrices. In general, order-N tensors are lists of order-$(N-1)$ tensors. Each of the N axis of an order-N tensor is referred to as a mode of the tensor. We denote tensors with uppercase Euler script ($\mathcal{X}$), matrices with

uppercase bold face ($\boldsymbol{X}$), vectors with lowercase bold face script ($\boldsymbol{x}$) and scalars with lower case script (x). We denote a specific element of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \ldots \times I_N}$ as $x_{i_1 i_2 \ldots i_N}$.

The mode-n matricization of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \ldots \times I_N}$ is the matrix formed by arranging all the vectors created by fixing every mode in $\mathcal{X}$ except for the nth mode as columns and is denoted as $\boldsymbol{X}_{(n)}$. Precisely,

$$\left(\boldsymbol{X}_{(n)}\right)_{i_n j} = x_{i_1 i_2 \ldots i_N} \text{ , where } j = 1 + \sum_{\substack{k=1 \\ k \neq n}}^{N} (i_k - 1) J_k \text{ and } J_k = \sum_{\substack{m=1 \\ m \neq n}}^{k-1} I_m \qquad (2.1)$$

An order-N tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \ldots \times I_N}$ is said to be of rank one if it can be written as the outer product between a set of n vectors, $\left\{\boldsymbol{a}^{(1)} \in \mathbb{R}^{I_1}, \boldsymbol{a}^{(2)} \in \mathbb{R}^{I_2}, \ldots, \boldsymbol{a}^{(N)} \in \mathbb{R}^{I_N}\right\}$. Concretely, $\mathcal{X} = \boldsymbol{a}^{(1)} \circ \boldsymbol{a}^{(2)} \circ \ldots \circ \boldsymbol{a}^{(N)}$. Here, $\circ$ denotes the vector outer product, defined as $x_{i_1 i_2 \ldots i_n} = a_{i_1}^{(1)} * a_{i_2}^{(2)} * \ldots * a_{i_N}^{(N)}$. The element-wise multiplication (also called Hadamard product) between equally sized tensors $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \ldots \times I_N}$ is denoted by the $*$ symbol and is defined as $(\mathcal{X} * \mathcal{Y})_{i_1 i_2 \ldots i_N} = x_{i_1 i_2 \ldots i_N} * y_{i_1 i_2 \ldots i_N}$. The norm of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \ldots \times I_N}$ is denoted as $\|\mathcal{X}\|$ and defined as $\|\mathcal{X}\| = \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \ldots \sum_{i_N=1}^{I_N} x_{i_1 i_2 \ldots i_N}^2}$.

$\boldsymbol{X}^{\dagger}$ denote the Moore-Penrose pseudo-inverse [48], of the matrix $\boldsymbol{X}$. Given matrices $\boldsymbol{A} \in \mathbb{R}^{I \times K}$ and $\boldsymbol{B} \in \mathbb{R}^{J \times K}$, $\boldsymbol{A} \odot \boldsymbol{B}$ denotes the Khatri-Rao product, resulting in a matrix belonging to $\mathbb{R}^{IJ \times K}$. Given a set of N matrices, $\left\{\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}\right\}$, $\boldsymbol{A}^{(-n)}$ denotes $\boldsymbol{A}^{(N)} \odot \ldots \odot \boldsymbol{A}^{(n+1)} \odot \boldsymbol{A}^{(n-1)} \odot \ldots \odot \boldsymbol{A}^{(1)}$.

2.2.2 Tensor Decomposition Algorithms

2.2.2.1 CANDECOMP/PARAFAC Decomposition (CPD)

TD is a class of algorithms that aim to compress a tensor into a set of factors, one for each mode of the tensor. One of the most popular TD algorithms is CPD. In essence, it seeks to decompose a tensor into a sum of rank-1 tensors. More concretely, given a tensor, $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ we wish to find three factor matrices, $\mathbb{C} = \left\{ \boldsymbol{C}^{(1)}, \boldsymbol{C}^{(2)}, \boldsymbol{C}^{(3)} \right\}$, where $\boldsymbol{C}^{(n)} \in \mathbb{R}^{I_n \times R}$ and R is a hyperparameter, such that the following holds:

$$\mathcal{X} = [\![\mathbb{C}]\!] \equiv \sum_{r=1}^{R} \boldsymbol{c}_{*r}^{(1)} \circ \boldsymbol{c}_{*r}^{(2)} \circ \boldsymbol{c}_{*r}^{(3)} \tag{2.2}$$

This is typically found by minimizing the following objective using the optimization algorithm called alternating least squares, for which more details can be found in Kolda and Bader [10]:

$$\mathbb{C}^* = \underset{\mathbb{C}}{argmin} \frac{1}{2} \|\mathcal{X} - [\![\mathbb{C}]\!]\|^2 \tag{2.3}$$

CP can easily be generalized to tensors of order greater or lesser than three.

2.2.2.2 CP-WOPT

Missing data is near ubiquitous in many applications of TD. However, CP is unable to cope with it. Acar et al. [13] remedy this by instead minimizing the following weighted objective:

$$f_{\mathcal{W}}(\mathbb{C}) = \frac{1}{2} \|\mathcal{W} * (\mathcal{X} - [\![\mathbb{C}]\!].)\|^2, \tag{2.4}$$

where $w_{ijk} = 1$ if x_{ijk} is observed and zero otherwise. This optimization problem is typically solved using first-order algorithms, such as nonlinear conjugate gradient or gradient descent [49], [50].

2.2.2.3 Deep Tensor Factorization

Presume that, in addition to $\mathcal{X}$, we have a tensor, $\mathcal{Y}$, which has the same size as $\mathcal{X}$. Further assume that there is a potentially complex and non-linear, functional mapping from the entries of $\mathcal{X}$ to that of $\mathcal{Y}$. Intuitively, $\mathcal{Y}$ can be viewed as the labels of $\mathcal{X}$.

Sun et al. [4] proposed the DTF algorithm to model this functional mapping using a MLP and CP-WOPT. Concretely, CP-WOPT is applied to $\mathcal{X}$ to obtain a set of factor matrices, $\mathbb{C}$, where $\mathbf{C}^{(\mathrm{n})} \in \mathbb{R}^{I_n \times R}$. Then, to predict y_{ijk}, $\mathbf{c}_{i*}^{(1)}$, $\mathbf{c}_{j*}^{(2)}$ and $\mathbf{c}_{k*}^{(3)}$ are used as inputs to an MLP. This was used to remarkable success in ACDSP and is depicted in **Figure 2.1**.

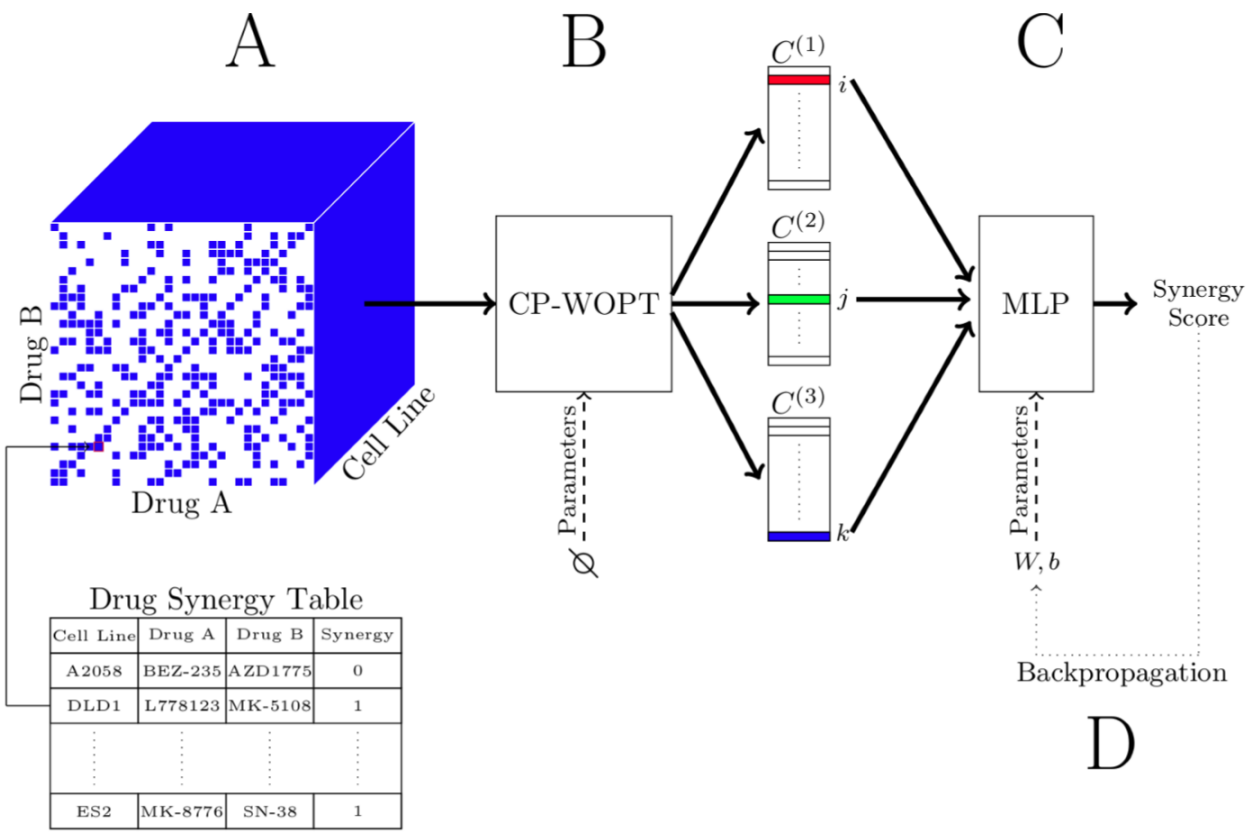

Figure 2.1 The workflow of the previous DTF algorithm. (A) Data collection and preprocessing. *(A) Drug-drug synergy (DDS) data was collected to construct a tensor. The synergy values were binarized to create synergy status, serving as labels to train the MLP. (B) The tensor is decomposed into a set of three factor matrices,* $\mathbb{C}$*, using CP-WOPT. (C) To predict the synergy score of the* $(i, j, k)^{th}$ *component of the DDS tensor, we input the* i^{th}*,* j^{th} *and* k^{th} *rows of the respective factor matrices into the MLP. (D) We update the weights and biases of the MLP using standard backpropagation. Note that CP-WOPT and the MLP are not trained in an end-to-end fashion.*

2.2.3 SCP: Selective CANDECOMP/PARAFAC

In Sun et. al's DTF algorithm, CP-WOPT attempts to compress all the information in the tensor into $\mathbb{C}$. These factors are then used to predict the labels. The result is that much of the information stored in $\mathbb{C}$ is completely irrelevant for classification. Due to the Curse-of-Dimensionality [46], this potentially has a detrimental effect on the classification performance. Thus, being able to selectively compress information in the tensor such that the factors only contain information that is directly relevant for classification may greatly increase accuracy.

To achieve this aim, we modify the CPD algorithm to include a second set of matrices, $\mathbb{P} = \left\{\mathbf{P}^{(1)}, \mathbf{P}^{(2)}, \mathbf{P}^{(3)}\right\}$. These matrices are the exact same size as the factor matrices and are referred to as parameter matrices. The motivation behind the parameter matrices is for them to contain the patterns that are relevant to the classification of the tensor, whereas the factor matrices will contain the prevalence of these patterns in the tensor.

Concretely, to decompose $\mathcal{X}$ into the set of factor matrices $\mathbb{C}$ while keeping $\mathbb{P}$ fixed, we compute the following:

$$\boldsymbol{C}^{(n)^*} = \underset{\boldsymbol{C}^{(n)}}{argmin} \frac{1}{2} \left\| \mathcal{X} - [\![\mathbb{P}/\mathbf{P}^{(\mathrm{n})} \cup \mathbf{C}^{(\mathrm{n})}]\!] \right\|^2 = \boldsymbol{X}_{(n)} \left[\mathbf{P}^{(-\mathrm{n})^T} \right]^{\dagger} \quad (2.5)$$

Computing the pseudo-inverse of $\mathbf{P}^{(-\mathrm{n})}$ can be inordinately expensive. The below alternative formulation, described in Kolda and Bader [10], deftly avoids this issue:

$$\boldsymbol{C}^{(n)^*} = \boldsymbol{X}_{(n)} \, \mathbf{P}^{(-\mathrm{n})} \mathbf{V}^{(\mathrm{n})^{\dagger}}, \text{ where } \mathbf{V}^{(\mathrm{n})} = \prod_{\substack{i=1 \\ i \neq n}}^{3} \mathbf{P}^{(i)^T} \mathbf{P}^{(i)} \quad (2.6)$$

Note that the Π operator computes the element-wise product and not matrix multiplication. We name the result Selective CANDECOMP/PARAFAC (SCP) and depict it in

Figure 2.2. SCP has the tremendous advantage of admitting a closed form calculation. SCP assumes that we are given a set of parameter matrices which already correspond to the patterns in the tensor that are useful for classification. In practice, they will start off as random and be iteratively updated until they converge to a useful value. This process is described in more detail in the following section.

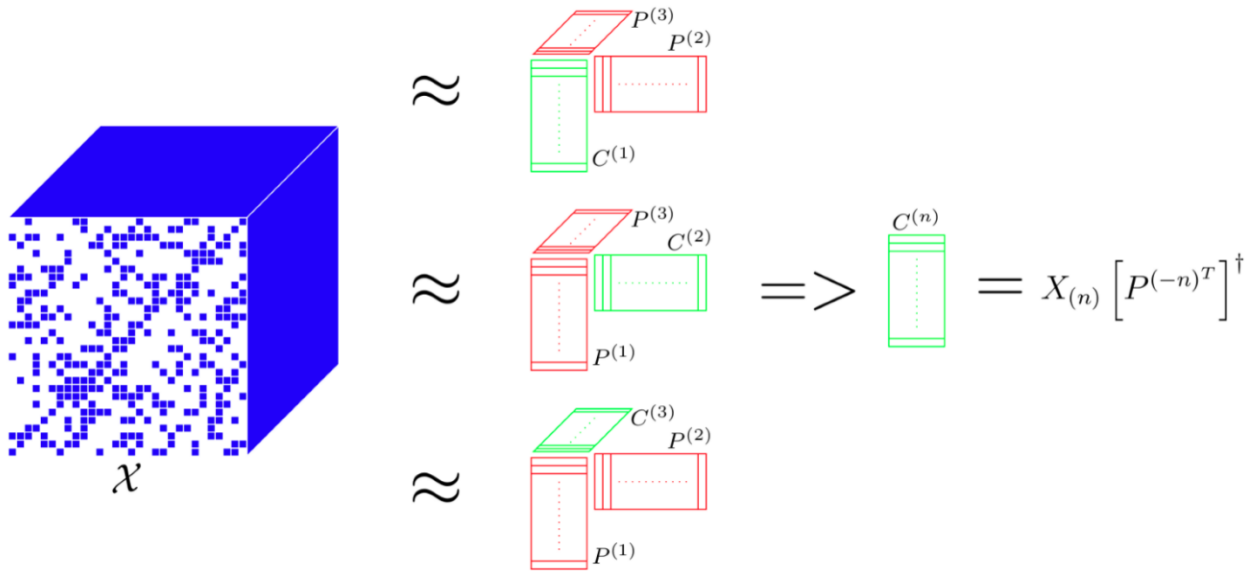

Figure 2.2 An illustration of SCP. *$\mathcal{X}$ is a tensor belonging to $\mathbb{R}^{I_1 \times I_2 \times I_3}$. $\mathbb{P} = \{\boldsymbol{P}^{(1)}, \boldsymbol{P}^{(2)}, \boldsymbol{P}^{(3)}\}$ are matrices belonging $\mathbb{R}^{I_n \times R}$, where R is a hyperparameter. $\mathbb{P}$ acts as parameters to SCP. Conceptually, they contain the features that are relevant for classification. $\mathbb{C}$ is the output of SCP and is the same size as $\mathbb{P}$. Conceptually, they indicate the prevalence of the features contained in $\mathbb{P}$. SCP aims to find the values for $\mathbb{C}$ that best reconstructs $\mathcal{X}$. This has the simple closed–form solution shown in the right-most equation. Red matrices indicate that the matrix is fixed parameter, green matrices indicate that the matrix is a free variable.*

2.2.4 Deep Neural Network

Like CP-WOPT, SCP outputs a set of factor matrices, $\mathbb{C}$. Similar to DTF we will use the factors from SCP as input to an MLP to predict the synergy scores for the entries of the tensor. Because SCP has a closed form calculation for matrix in $\mathbb{C}$ that depends on $\mathbb{P}$ we can easily treat the latter as a trainable parameter. This allows for $\mathbb{P}$ to be trained alongside the parameters of the MLP in an end-to-end fashion. Concretely, we optimize the following:

$$\mathbb{P}^*, \mathbb{W}^* = \underset{\mathbb{P},\mathbb{W}}{argmin} \frac{1}{2} \left\| \mathcal{W} * (\mathcal{Y} - \tilde{\mathcal{Y}}) \right\|^2, \quad (2.7)$$

where $\mathcal{Y} = MLP(SCP(\mathcal{X}, \mathcal{W}; \mathbb{P}); \mathbb{W})$, and $\mathbb{W}$ are the weights of the MLP. The solutions to this objective can easily be found with any stochastic gradient descent algorithm [50]. The gradient of both the weights and parameter matrices can automatically be calculated using modern DL frameworks such as TensorFlow [51] or PyTorch [52].

The result is Deep Selective CANDECOMP/PARAFAC (DSCP), an algorithm that can automatically find and select patterns in a tensor that are relevant for classification and train a classifier using them. DSCP is depicted in **Figure 2.3** and its details are shown in Algorithm 1.

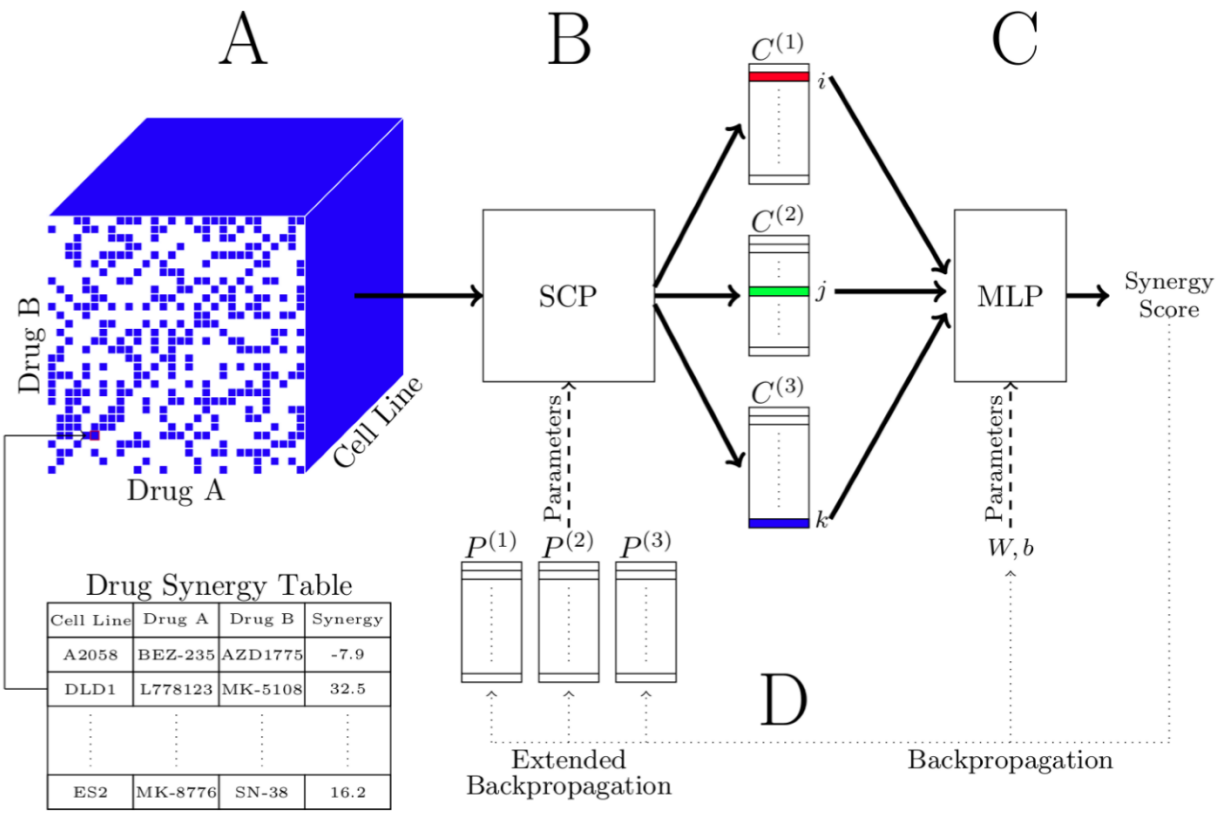

Figure 2.3 The workflow of the proposed DSCP algorithm. *(A) Data collection and preprocessing. Drug-drug synergy (DDS) data was collected to construct a tensor. (B) The features within the DDS tensor relevant for the classification of synergistic drugs are decomposed into three factor matrices,* $\boldsymbol{C}^{(1)}$, $\boldsymbol{C}^{(2)}$ *and* $\boldsymbol{C}^{(3)}$, *using SCP. (C) To predict the synergy score of the* $(i, j, k)^{th}$ *component of the DDS tensor, we input the* i^{th}, j^{th} *and* k^{th}*rows of the respective factor matrices into the MLP. (D) We update the weights and biases of the MLP using standard backpropagation and update the parameters matrices of SCP using extended backpropagation. Note that this is done in an end-to-end fashion.*

Algorithm 1 Deep Selective CANDECOMP/PARAFAC

function SCP($\mathcal{X}, \mathcal{W}; \mathbb{P}$)
  $\bar{\mathcal{X}} \leftarrow \mathcal{X} * \mathcal{W}$;
  for $n \leftarrow 1$ to 3 **do**
    $\bar{\mathbf{X}}_{(n)} \leftarrow Matricize(\bar{\mathcal{X}}, n)$;
    $\mathbf{V}^{(n)} \leftarrow \prod_{\substack{i=1 \\ i \neq n}}^{3} \mathbf{P}^{(n)T} \mathbf{P}^{(n)}$;
    $\mathbf{P}^{(-n)} \leftarrow \mathbf{P}^{(N)} \odot ... \odot \mathbf{P}^{(n+1)} \odot \mathbf{P}^{(n-1)} \odot ... \odot \mathbf{P}^{(1)}$;
    $\mathbf{C}^{(n)} \leftarrow \mathbf{X}_{(n)} \mathbf{P}^{(-n)} \mathbf{V}^{(n)^{\dagger}}$;
  end for
  return $\{\mathbf{C}^{(1)}, \mathbf{C}^{(1)}, \mathbf{C}^{(3)}\}$
end function
function TEST($\mathcal{X}, \mathcal{W}; \mathbb{W}, \mathbb{P}\}$)
  $\mathbb{I} \leftarrow Size_Of(\mathcal{X})$;
  $\tilde{\mathcal{Y}} \leftarrow Zeros(\mathbb{I})$;
  $\mathbb{C} \leftarrow SCP(\mathcal{X}, \mathcal{W}; \mathbb{P})$;
  for $i_1, i_2, i_3 \leftarrow 1$ to I_1, I_2, I_3 **do**
    $\tilde{\mathcal{Y}}_{i_1,...,i_n} \leftarrow MLP(\{\mathbf{C}^{(1)}_{i_1*}, ..., \mathbf{C}^{(n)}_{i_n*}\}; \mathbb{W})$;
  end for
  return $\tilde{\mathcal{Y}}$
end function
function TRAIN($\mathcal{X}, \mathcal{W}, \mathcal{Y}, R$)
  $\mathbb{I} \leftarrow Size_Of(\mathcal{X})$;
  $\mathbb{P} \leftarrow Init_Random(\mathbb{I}, R)$;
  $\mathbb{W} \leftarrow MLP_Init()$;
  for *epoch* $\leftarrow 1$ to *maxepochs* **do**
    for *mini_batch_indices* in *mini_batches* **do**
      $\mathbb{C} \leftarrow SCP(\mathcal{X}, \mathcal{W}; \mathbb{P})$;
      $\nabla\mathbb{W}, \nabla\mathbb{P} \leftarrow 0$;
      for i in *mini_batch_indices* **do**
        $\tilde{y} \leftarrow MLP(\{\mathbf{C}^{(1)}_{i_1*}, ..., \mathbf{C}^{(n)}_{i_n*}\}; \mathbb{W})$;
        $\tilde{e} \leftarrow Loss(y_{i_1,i_2,i_3}, \tilde{y}) * w_{i_1,i_2,i_3}$;
        $\nabla\mathbb{W}, \nabla\mathbb{P} \leftarrow \nabla\mathbb{W}, \nabla\mathbb{P} + Backprop(\tilde{e}, \mathbb{W}, \mathbb{C}, \mathbb{P})$;
      end for
      $\mathbb{W}, \mathbb{P} \leftarrow ADAM(\nabla\mathbb{W}, \nabla\mathbb{P})$;
    end for
  end for
  return $\mathbb{W}, \mathbb{P}$
end function

2.2.5 Hyperparameter Description

What follows is a description of DSCP's hyperparameters.

- Rank: The number of columns in both the parameter and factor matrices. This hyperparameter influences the expressivity of DSCP, with higher values allowing for more expressivity and lower values having a regularization effect.
- MLP Architecture: The size of each layer in the MLP. The format follows [Layer 1 Size Layer 2 Size … Layer N Size].
- Hidden Layers DR: The dropout rate of each hidden layer of the MLP.
- Input Layers DR: The dropout rate of the output of the first layer of the MLP.
- Batch Size: The batch size used in the stochastic gradient descent algorithm.

2.3 Data Sources and Algorithm Evaluation Strategy

Table 2.1 A summary of the algorithms, data sources and cross-validation strategies considered in this chapter.

Algorithms
DTF [4], DeepSynergy[35], CoSTCo [18], CP-WOPT [13], PRODeepSyn [37], Average, DSCP
Data Sources
Merick and Co [2], NCI-ALMANAC [47], Synthetic Tensor
Cross-Validation Strategies
Leave-Drug-Combination-Out (LDCO), Leave-Cell-Line-Out (LCLO), Leave-Drug-Out (LDO)
Metrics
ROC-AUC, PR-AUC, Precision, Recall, F1-Score, BACC, Kappa, R2, MAE, MDAE, RMSE
Software
TensorFlow (2.4.1)[51], Tensorly (0.6.0) [63], SciPy (1.7.3) [64], Scikit-Learn (0.24.2) [58]

2.3.1 Data Sources

2.3.1.1 Drug Combination Compound Screens

To evaluate DSCP, we utilized two anticancer drug synergy compound screens. The Merick and Co compound screen [2] contains 38 drugs and 39 cell lines from 7 tissue types. This

results in 28’000 possible drug and cell line combinations, however only 23’000 of these were gathered by the original authors. Each of these combinations were tested *in vitro,* with the Loewe Additivity metric [53] being recorded. A negative Loewe Additivity metric indicates that the drugs are anti-synergistic, a positive value indicates that they are synergistic, and a zero value indicates that they are additive. Following previous literature [3], [35], [36] we consider combinations with a Loewe Additivity metric higher than 30 to be highly synergistic.

The National Cancer Institute (NCI)-ALMANAC [47] compound screen contains 104 drugs and 60 cell lines from 10 tissue types. This results in 321’000 possible drug and cell line combinations, however only 300’000 of these were gathered by the original authors. Each of these combinations was evaluated *in vitro*, with the ComboScore [47] being recorded. A negative ComboScore indicates that the drugs are anti-synergistic, a positive value indicates that they are synergistic, and a zero value indicates that they are additive. Following previous literature [3] we consider combinations with a ComboScore higher than 10 to be highly synergistic.

For both compound screens, the ACDSP task can be treated as either regression or classification. For regression, the raw synergy value, either Loewe or ComboScore, is predicted. For classification, highly synergistic drug pairs are treated as the positive label and all others are treated as negative labels.

2.3.1.2 Drug and Cell Line Auxiliary Features

DSCP utilizes the drug features outlined in DeepSynergy’s original publication. They use three sets of features for the drugs: Counts of extended connectivity fingerprints, physicochemical properties and binary toxicophores. The authors of DeepSynergy provide this data for the Merick and Co compound screen, but not for DrugComb or NCI-ALMANAC. For these datasets, we follow their methodology for generating the drug features as closely as

possible. The software package ChemoPy [54] was used to generate the physicochemical properties. Because ChemoPy has not been updated in many years, it was only possible to extract 540 out of the 830 properties that the authors used. The web server that was used to generate the Binary Toxophores is no longer functional and thus it was not possible to generate these features. RDKit [55] was used to generate the counts of extended connectivity fingerprints.

Once again, we follow the procedure used by the authors of DeepSynergy to gather cell line gene expression data. We first downloaded the genomic data from the ArrayExpress database [56] and quantile normalized it. Finally, we used the FARMS software package [57] to filter out non-informative genes.

2.3.1.3 Synthetic Data Generation

To evaluate the efficacy of DSCP's ability to be trained end-to-end, we compared it to a non-end-to-end trainable version. To accomplish this, we utilized a synthetic tensor classification dataset. The tensor's size was 6400 X 50 X 50 X 50 and had rank 10. The information relevant to classification had only a rank of 2 and the information irrelevant to classification had a rank of 8. The procedure used to generate the data is outlined below:

- Randomly generate 3, 50 X 10 factor matrices with integer entries, uniformly distributed within [-100,100]
- Use Scikit-Learn [58] to generate a binary classification dataset with 10 features--two useful and eight irrelevant--and 6400 data points.
- Convert each of these vectors into a factor matrix of size 6400 X 10
- Generate the final tensor by treating these matrices as the factors of a CP decomposition.
- Treat the labels as a tensor of size 6400.

2.3.2 Cross-Validation Strategy

Several different approaches, depicted in **Figure 2.4**, to separating the data into training and testing sets can be taken. First, one can randomly separate each drug-drug-cell line combination into train and test splits. This has been shown to inflate performance as compound screens do not choose which combinations to evaluate randomly. An alternative approach, called Leave-Drug-Combination-Out (LDCO), is to randomly separate each drug-drug pair across all cell lines. This is more realistic, as this is typically how compound screens are executed. There exists two other, more strict, cross-validation strategies, Leave-Drug-Out (LDO) and Leave-Cell-Line-Out (LCLO), where all the data for a drug or cell line is put in exactly one of the data splits. This is a far harder problem, as the algorithms have no previous interactions for the drugs and cell lines in the test set to base their predictions one. For the Merick and Co compound screen, the LCDO, LDO and LCLO strategies were used. For the NCI-ALMANAC compound screen, only LDCO was used as NCI-ALMANAC is a much larger dataset and requires significantly more computational resources.

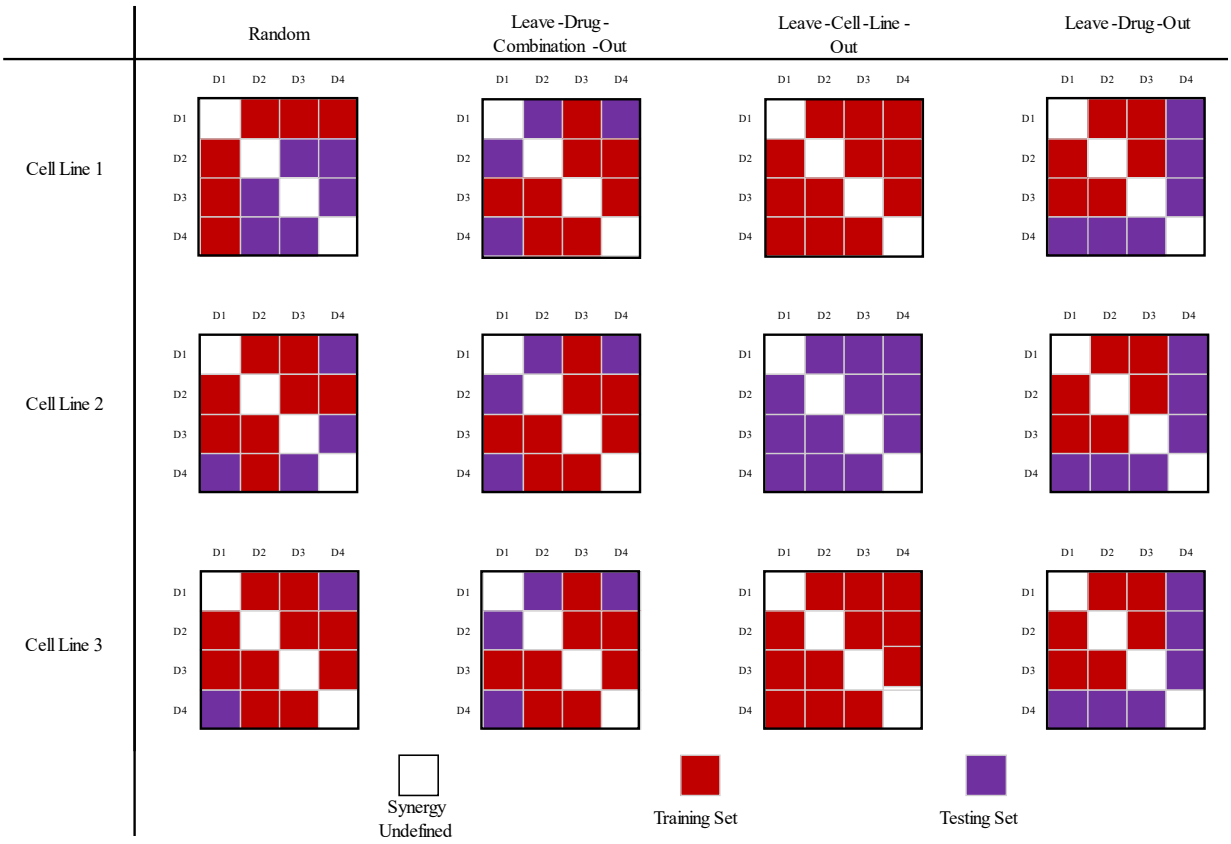

Figure 2.4 A depiction of the various cross-validation strategies that can be employed for ACDSP.

2.3.3 Compared Algorithms

We compared the performance of our proposed algorithms to many ACDSP algorithms. These algorithms come in two main categories: those based on TC and those on a traditional DL framework.

Average: A baseline algorithm. This algorithm simply averages drug synergy results from the training set and extrapolates it to the testing set. For the LCLO task, the average of each drug pair in the training set and that average is extrapolated to the cell lines in the testing set. For the LDO task, the average of each drug-cell line pair in the training set and is extrapolated to the

drugs in the testing set. There is no clear way to extrapolate an average value on the LDCO task, thus we did not evaluate Average on this task.

CANDECOMP/PARAFAC-Weighted Optimization (CP-WOPT) [13]: A linear TC algorithm. This algorithm finds a low-rank CPD [11] for the observed values in a partially observed tensor. This decomposition is then used to impute the missing values of the tensor. In the context of ACDSP, these missing values are the drug combinations not examined in the compound screen.

Convolutional Sparse Tensor Completion (CoSTCo) [18]: A convolutional-based DTC algorithm. This algorithm finds a set of factor matrices that, when used as input into a shallow convolutional network, can reconstruct the observed entries of a partially observed tensor. As with CP-WOPT, CoSTCo can then be used to impute the missing values of the tensor.

Deep Tensor Factorization (DTF) [4]: A DTC algorithm based on CP-WOPT and an MLP. It first executes CP-WOPT on a partially observed tensor without the low-rank assumption. Then, using the decomposed tensor as features, it uses an MLP to predict the synergies between pairs of drugs.

DeepSynergy [35]: An MLP-based synergy prediction algorithm. DeepSynergy uses the physiochemical and genomic properties of the drugs and cell lines respectively to train an MLP to predict the synergies between drug pairs.

PRODeepSyn [37]: An MLP-based synergy prediction algorithm. PRODeepSyn uses the physiochemical properties of the drugs and extracts properties from the cell lines' protein-protein interaction network using a graph convolutional network. Using these features, an MLP is trained to predict synergies between drug pairs.

HyperGraphSynergy [40]: An algorithm based on a hypergraph neural network and an MLP. It forms a hypergraph by representing drugs and cell lines as nodes and synergies as hyperedges. A hypergraph neural network is then used to extract features for the cell lines and drugs followed by an MLP to predict synergies.

2.3.4 Metrics

As discussed in section 2.3.1, we can treat drug synergy prediction as a classification task or a regression task. For this reason, we use both classification and regression metrics to evaluate the performance of the algorithms. When treating the problem as a regression we use the following metrics:

- Coefficients of Determination: This metric ranges from -1 to 1. A value of 0 indicates random predictions, a negative value indicates worse than random, and a positive value indicates better than random. A higher value is better. It is defined below:

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y_i})^2}{\sum_{i=1}^{n}(y_i - \bar{y_i})^2} \quad (2.8)$$

  where $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ and n is the number of samples.

- Mean Absolute Error: This ranges from 0 to positive infinity with lower values being better. It is defined below:

$$MAE(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y_i}| \quad (2.9)$$

- Median Absolute Error: This metric ranges from 0 to positive infinity and is more robust against outliers than the MAE. Lower values are better, and it is defined below:

$$MDAE(y, \hat{y}) = median(|y_1 - \widehat{y_1}|, \dots, |y_n - \widehat{y_n}|) \quad (2.10)$$

- Root Mean Squared Error: This metric ranges from 0 to positive infinity and is more sensitive to outliers than both MAE and MDAE. Lower values are better, and it is defined below:

$$RMSE(y, \hat{y}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2} \quad (2.11)$$

Before we define the classification metrics used, we must define the following:

- True Positive (tp): The number of positively labelled samples that the algorithm correctly identifies as positive.
- True Negative (tn): The number of negatively labelled samples that the algorithm correctly identifies as negative.
- False Positive (fp): The number of negatively labelled samples that the algorithm incorrectly identifies as positive.
- False Negative (fn): The number of positively labelled samples that the algorithm incorrectly identifies as negative.

In this thesis, we use the following classification metrics:

- Precision: This is a measure of how conservative an algorithm is at predicting positive samples. It ranges between 0 and 1, with higher values indicating less false positive predictions. It is defined below:

$$prec = \frac{tp}{tp + fp} \quad (2.12)$$

- Recall: This is a measure of how likely an algorithm is to correctly label positive samples as such. It ranges between 0 and 1 with higher values indicating more true positive predictions. It is defined below:

$$recall = \frac{tp}{tp + fn} \quad (2.13)$$

- F1-Score: This is a comprehensive metric that encompasses both precision and recall. It ranges between 0 and 1 with higher values indicating higher underlying precision and recall values and thus better predictions. It is defined below:

$$F_1 = \frac{2 \times prec \times recall}{prec \times recall} \quad (2.14)$$

- Cohen's Kappa: This is a comprehensive metric that is robust to class imbalance. It ranges between -1 and 1, with higher values being better. It is defined below:

$$kappa = \frac{2 \times (tp \times tn - fn \times fp)}{(tp + fp) \times (fp + tn) \times (tp + fn) \times (fn + tn)} \quad (2.15)$$

- Balanced Accuracy: This is a variant of accuracy that accounts for class imbalance. It ranges between 0 and 1, with higher values being better. It is defined below:

$$bacc = \frac{1}{2}\left(\frac{tp}{tp + fn} + \frac{tn}{tn + fp}\right) \quad (2.16)$$

- Area Under Precision Recall Curve (PR-AUC): The precision and recall of an algorithm over a range of thresholds are used to create the precision-recall curve. Then the area under this curve is taken.
- Area Under Characteristic Receiving Curve (ROC-AUC): The true positive rate $\left(\frac{tp}{n}\right)$ and false positive rate $\left(\frac{fp}{n}\right)$ of an algorithm over a range of thresholds are used to create a characteristic receiving curve. Then the area under this curve is taken.

For the classification metrics not based on AUC, we must choose a classification threshold for the predictions of our algorithms. We do this by finding the threshold that maximizes Youden's J-Statistic [59], defined below:

$$J = \frac{tp}{tp + fn} + \frac{tn}{tn + fp} - 1 \quad (2.17)$$

2.3.5 Hyperparameter Optimization

Each algorithm has several important hyperparameters that can greatly affect their performance. For this reason, we perform an extensive hyperparameter search for each algorithm. We use the Bayesian optimization library Optuna (2.10.0) [60] over 100 trials. For the Merick and Co compound screen, we performed 5-fold cross-validation at each trial on the training set. For the NCI-ALMANAC compound screen we simply used 20% of the training set as a validation set. This choice was due to the immense size of the NCI-ALMANAC compound screen. Early stopping based on the training loss was used and the learning rate was automatically reduced on plateaus in the loss.

Additionally, we use Optuna to generate an estimate of the importance of each algorithm's hyperparameters. Specifically, a random forest algorithm [61] is trained to predict the algorithm's predictive performance using its hyperparameters. Then, functional ANOVA [62] is applied to estimate which hyperparameters best predicts the algorithm's performance.

2.3.6 Test Set Inference

For the final evaluation, the entire training set was used to train each algorithm with their optimal hyperparameters. The predictions of the algorithm on the testing set were saved and the

performance metrics recorded. This final evaluation was repeated 10 times for each algorithm, with the mean and standard deviation of the 10 trials being reported. We compared the performance of our proposed algorithms regarding a particular metric to the performance of the published algorithms using a two-sided independent students t-test at the 95% confidence interval.

2.3.7 Software and Hardware

If a public GitHub repository was not available, the algorithms were implemented using TensorFlow (2.4.1)[51] . The tensor-based algorithms used a package for tensor arithmetic called Tensorly (0.6.0) [63]. SciPy (1.7.3) [64] and Scikit-Learn (0.24.2) [58] were used to calculate the performance metrics. The experiments were executed on a machine with an RTX 2080 Ti GPU, an AMD Ryzen Threadripper 3960x 24-core processor and 512 gigabytes of DDR4 RAM.

2.4 Results

2.4.1 Synthetic Data Analysis

DTF and DSCP are similar algorithms in that they both use a CP-based decomposition for feature extraction and an MLP for synergy prediction. The primary difference is that DSCP is trainable end-to-end and can selectively decompose information, whereas DTF can do neither. For this reason, we sought to compare the two algorithms in a controlled setting analyzing the effect that these two differences have on performance. This was accomplished by comparing DTF and DSCP on the synthetic tensor outlined in section 2.3.1.3.

We evaluated DTF and DSCP with ranks 5, 10, 15, 20, 30, 40, and 50 on the fully observed synthetic tensor. We employed a 90%/10% train/test split and report the test ROC-AUC. We ran each algorithm five times and reported the box plot. The results are shown in **Figure 2.5**. From this, we can see that DSCP is less sensitive to the chosen rank then DTF and is still able to find the correct factors at rank 5. Additionally, DSCP simply maintained a higher ROC-AUC at each of the ranks. We evaluated DTF and DSCP, with rank 10 on the synthetic tensor with a random percentage of entries removed. The percentages 0%, 50%, 75% and 90% were evaluated and the results are shown in **Figure 2.6**. From this, we can see that both are able to cope with substantial amounts of missing data with little or no drop in performance. Finally, we evaluate both algorithms with 50%, 75%, 90% and 95% class imbalance. We use the PR-AUC metric as it is a powerful metric for classification with class imbalance. The results can be seen in **Figure 2.7**. From this, we can see that DSCP maintains its advantage even under severe class imbalance.

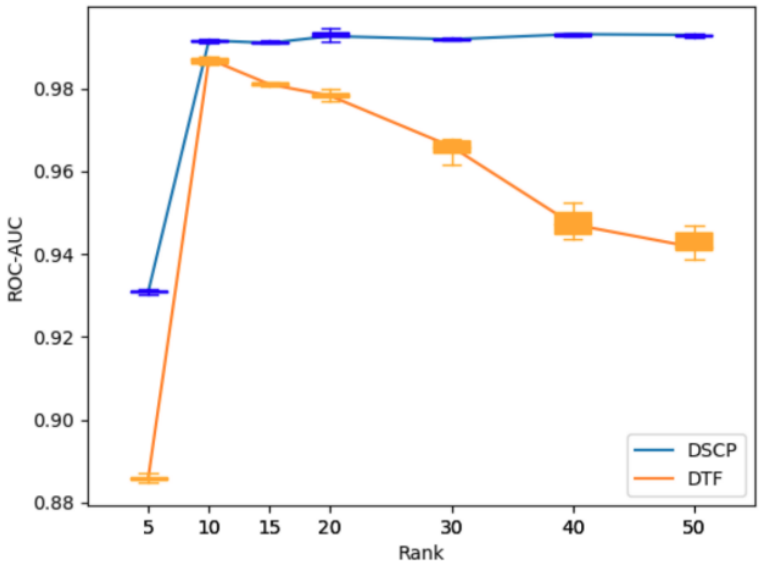

Figure 2.5 The ROC-AUC of the DTF and DSCP algorithms, with varying rank, on the synthetic tensor task.

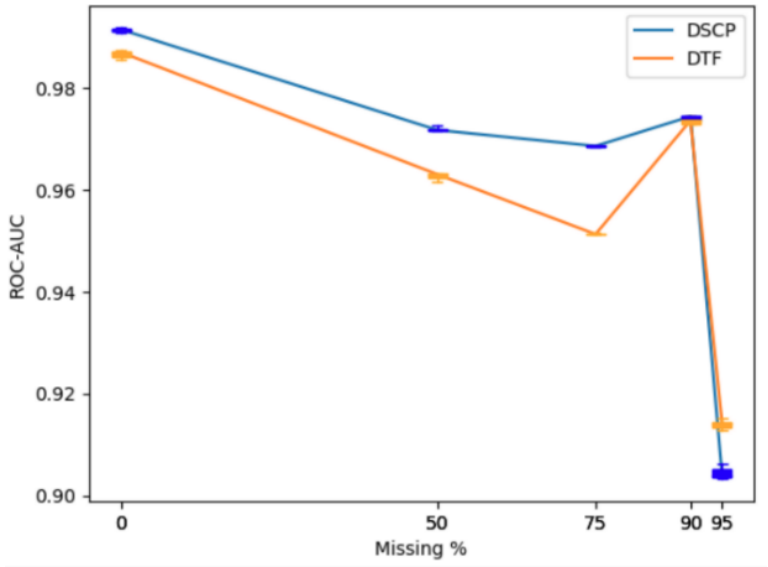

Figure 2.6 The ROC-AUC of the DTF and DSCP algorithms on the synthetic tensor task with varying levels of missing data.

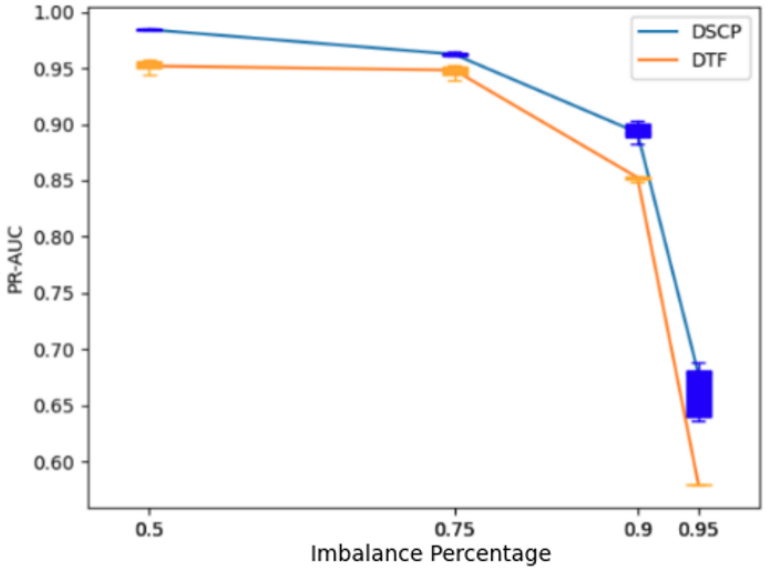

Figure 2.7 The PR-AUC of the DTF and DSCP algorithms, with varying class imbalance, on the synthetic tensor task.

2.4.2 Merick and Co Compound Screen

2.4.2.1 Leave-Drug-Combination-Out (LDCO)

Table 2.2 The results of the hyperparameter search for DSCP on the Merick and Co compound screen using the LDCO cross-validation strategy. *DR stands for Dropout Rate.*

Hyperparameter	Considered Values	Chosen Value	Importance
Rank	1-256	224	8%
MLP Architecture	[1024 1024 512], [2048 2048 1024], [2048 1024 512], [2048 2048 2048], [1024 1024 1024], [2048 2048], [1024 1024], [512 512], [1024 512]	[1024 512]	56%
Hidden Layers DR	0-0.5	0.5	7%
Input Layers DR	0-0.2	0.1	26%
Batch Size	512, 1024, 2048	1024	4%

From the results of the hyperparameter search (**Table 2.2**) we can see that, with 56% importance, the MLP architecture is the hyperparameter that best predicts the algorithms cross-validation performance. The architecture is chosen to be quite small. A high hidden layer DR (0.5) and small MLP architecture indicates that it was crucial for DSCP to avoid overfitting. On

the both the classification (**Table 2.3**) and regression (**Table 2.4**) task DSCP performs similarly to CoSTCo on most metrics. It performs better than CP-WOPT on most metrics except precision. Finally, on most metrics DSCP performs significantly worse than DTF, DeepSynergy and PRODeepSyn at the 95% confidence interval.

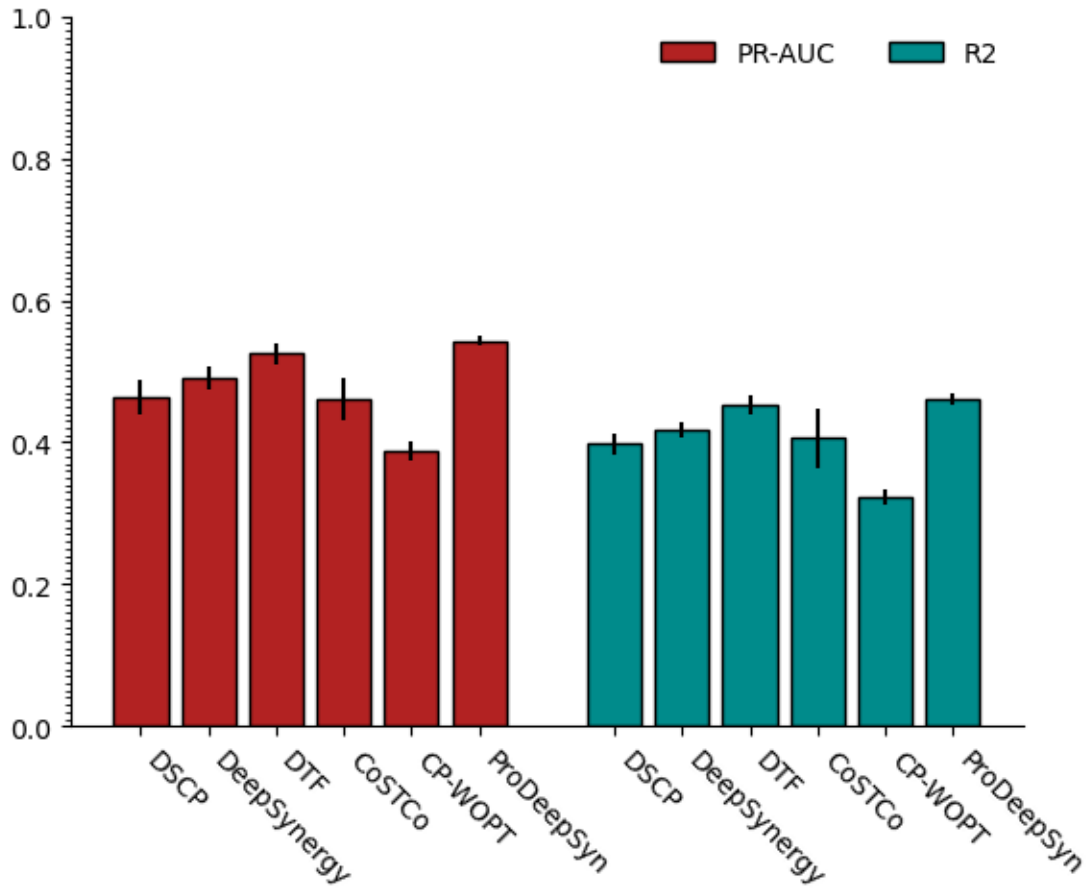

Figure 2.8 A summary of the classification and regression results on the Merick and Co compound screen using LDCO cross-validation.

Table 2.3 The classification results of our compared algorithms on the Merick and Co compound screen using LDCO cross-validation. *Red indicates that the algorithm performed worse than DSCP, green indicates better and black indicates that the result was insignificant. The statistics test used is outlined in section 2.3.6.*

Algorithm	ROC-AUC	PR-AUC	Recall	Prec.	F1	BACC	Kappa
DeepSynergy	0.84 ± 0.01	0.50 ± 0.02	0.54 ± 0.07	0.44 ± 0.08	0.47 ± 0.02	0.74 ± 0.02	0.42 ± 0.03
DTF	0.84 ± 0.01	0.52 ± 0.01	0.49 ± 0.04	0.51 ± 0.07	0.49 ± 0.02	0.72 ± 0.01	0.45 ± 0.02
CoSTCo	0.81 ± 0.02	0.46 ± 0.03	0.50 ± 0.05	0.40 ± 0.03	0.45 ± 0.02	0.72 ± 0.02	0.39 ± 0.03
CP-Wopt	0.79 ± 0.01	0.39 ± 0.01	0.41 ± 0.04	0.38 ± 0.03	0.39 ± 0.02	0.68 ± 0.02	0.34 ± 0.02
PRODeepSyn	0.85 ± 0.00	0.54 ± 0.01	0.42 ± 0.04	0.62 ± 0.05	0.49 ± 0.01	0.70 ± 0.02	0.46 ± 0.01
DSCP	0.82 ± 0.01	0.46 ± 0.02	0.50 ± 0.03	0.41 ± 0.04	0.45 ± 0.03	0.72 ± 0.01	0.39 ± 0.03

Table 2.4 The regression results of our compared algorithms on the Merick and Co compound screen using LDCO cross-validation. *Red indicates that the algorithm performed worse than DSCP, green indicates better and black indicates that the result was insignificant. The statistics test used is outlined in section 2.3.6.*

Algorithm	R^2	RMSE	MAE	MDAE
DeepSynergy	0.42 ± 0.02	16.86 ± 0.32	10.40 ± 0.23	6.93 ± 0.20
DTF	0.45 ± 0.01	16.45 ± 0.20	10.22 ± 0.19	6.78 ± 0.17
CoSTCo	0.41 ± 0.04	17.12 ± 0.60	11.28 ± 0.39	7.83 ± 0.29
CP-Wopt	0.32 ± 0.01	18.29 ± 0.15	11.97 ± 0.17	8.40 ± 0.16
PRODeepSyn	0.46 ± 0.01	16.32 ± 0.10	10.05 ± 0.10	6.70 ± 0.13
DSCP	0.40 ± 0.01	17.24 ± 0.21	10.93 ± 0.17	7.40 ± 0.15

2.4.2.2 Leave-Drug-Out (LDO)

Table 2.5 The results of the hyperparameter search for DSCP on the Merick and Co compound screen using the LDO cross-validation strategy.

Hyperparameter	Considered Values	Chosen Value	Importance
Rank	1-256	208	78%
MLP Architecture	[1024 1024 512], [2048 2048 1024], [2048 1024 512], [2048 2048 2048], [1024 1024 1024], [2048 2048], [1024 1024], [1024 512] [512 512],	[512 512]	8%
Hidden Layers DR	0-0.5	0.3	1%
Input Layers DR	0-0.2	0.2	5%
Batch Size	512, 1024, 2048	512	8%

From the results of the hyperparameter search (**Table 2.5**) we can clearly see that the most important hyperparameter is the rank. A high rank was found to be critical to the performance of DSCP. On the regression task (**Table 2.7**) no algorithm consistently outperforms the Average baseline. DSCP, CP-WOPT and CoSTCo appear to have failed to learn to make meaningful predictions, with R2 values close to or lower than zero. On both the regression (**Table 2.7**) and classification (**Table 2.6**) task DeepSynergy, DTF and PRODeepSyn all consistently outperform DSCP.

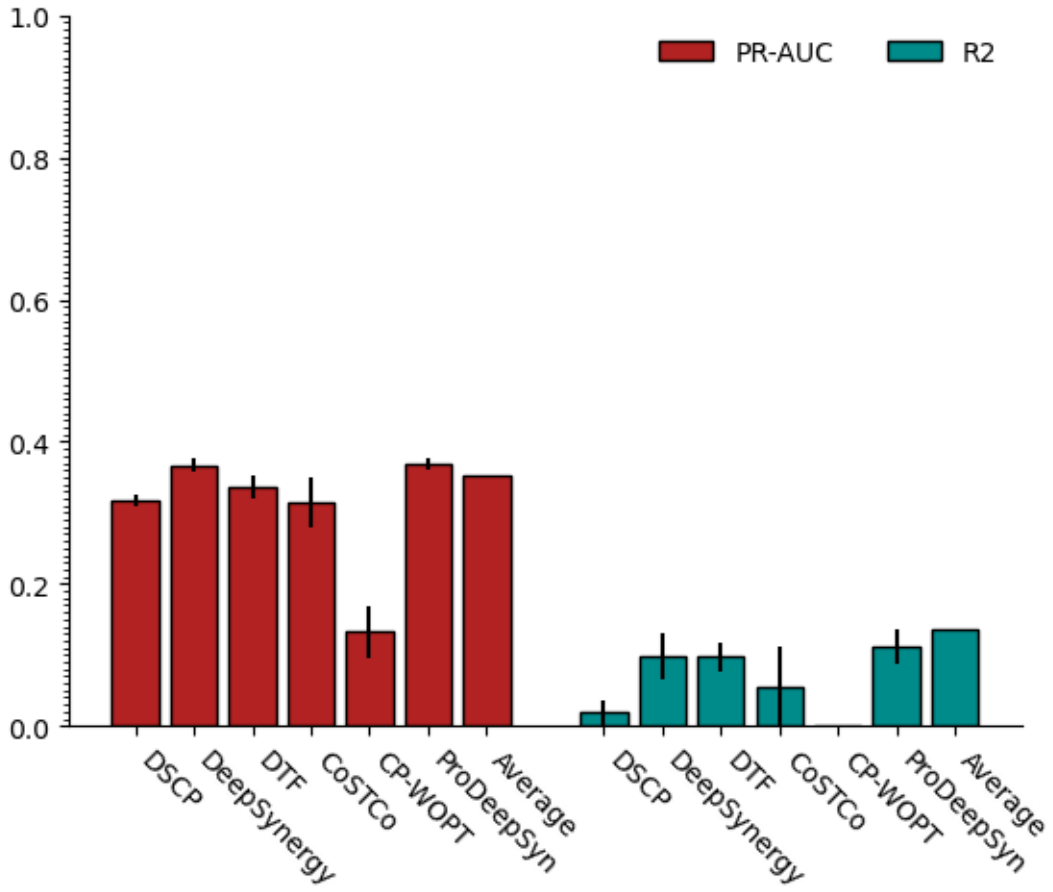

Figure 2.9 A summary of the classification and regression results on the Merick and Co compound screen using LDO cross-validation.

Table 2.6 The classification results of our compared algorithms on the Merick and Co compound screen using LDO cross-validation. *Red indicates that the algorithm performed worse than DSCP, green indicates better and black indicates that the result was insignificant. The statistics test used is outlined in section 2.3.6.*

Algorithm	ROC-AUC	PR-AUC	Recall	Prec.	F1	BACC	Kappa
Average	0.74 ± 0.00	0.35 ± 0.00	0.32 ± 0.00	0.41 ± 0.00	0.36 ± 0.00	0.63 ± 0.00	0.28 ± 0.00
DeepSynergy	0.74 ± 0.01	0.37 ± 0.01	0.30 ± 0.04	0.46 ± 0.03	0.36 ± 0.02	0.63 ± 0.01	0.30 ± 0.02
DTF	0.72 ± 0.01	0.34 ± 0.02	0.20 ± 0.05	0.48 ± 0.05	0.28 ± 0.04	0.59 ± 0.02	0.23 ± 0.03
CoSTCo	0.70 ± 0.03	0.31 ± 0.04	0.31 ± 0.07	0.38 ± 0.05	0.33 ± 0.05	0.62 ± 0.03	0.26 ± 0.04
CP-WOPT	0.51 ± 0.05	0.13 ± 0.04	0.25 ± 0.25	0.08 ± 0.05	0.11 ± 0.08	0.50 ± 0.01	0.00 ± 0.01
PRODeepSyn	0.74 ± 0.00	0.37 ± 0.01	0.27 ± 0.03	0.48 ± 0.02	0.35 ± 0.02	0.62 ± 0.01	0.29 ± 0.02
DSCP	0.69 ± 0.01	0.32 ± 0.01	0.12 ± 0.01	0.58 ± 0.03	0.20 ± 0.02	0.56 ± 0.01	0.17 ± 0.01

Table 2.7 The regression results of our compared algorithms on the Merick and Co compound screen using LDO cross-validation. *Red indicates that the algorithm performed worse than DSCP, green indicates better and black indicates that the result was insignificant. The statistics test used is outlined in section 2.3.6.*

Algorithm	R^2	RMSE	MAE	MDAE
Average	0.14 ± 0.00	22.18 ± 0.00	14.41 ± 0.00	9.94 ± 0.00
DeepSynergy	0.10 ± 0.03	22.64 ± 0.41	14.82 ± 0.42	10.25 ± 0.40
DTF	0.10 ± 0.02	22.67 ± 0.26	14.75 ± 0.22	10.01 ± 0.17
CoSTCo	0.05 ± 0.06	23.18 ± 0.68	15.41 ± 0.54	10.74 ± 0.47
CP-WOPT	-0.07 ± 0.00	24.65 ± 0.02	16.04 ± 0.01	10.89 ± 0.02
PRODeepSyn	0.11 ± 0.02	22.48 ± 0.30	14.72 ± 0.32	10.19 ± 0.31
DSCP	0.02 ± 0.02	23.63 ± 0.22	15.66 ± 0.22	10.79 ± 0.23

2.4.2.3 Leave-Cell-Line-Out (LCLO)

Table 2.8 The results of the hyperparameter search for DSCP on the Merick and Co compound screen using the LCLO cross-validation strategy.

Hyperparameter	Considered Values	Chosen Value	Importance
Rank	1-256	33	62%
MLP Architecture	[1024 1024 512], [2048 2048 1024], [2048 1024 512], [2048 2048 2048], [1024 1024 1024], [2048 2048], [1024 1024], [512 512], [1024 512]	[1024 1024]	22%
Hidden Layers DR	0-0.5	0.3	7%
Input Layers DR	0-0.2	0.0	2%
Batch Size	512, 1024, 2048	512	7%

Unlike the LDCO on LDO task, a low rank (33) is preferred with high importance (62%) on the LCLO task. The MLP architecture is the next most important parameter with 22% importance. This is shown in **Table 2.8**. On the classification (**Table 2.9**) task, DSCP outperforms DeepSynergy on most metrics and shows mixed results when compared to Average, DTF, CoSTCo and PRODeepSyn. It has a higher ROC-AUC, BACC and Recall than most other algorithms and a worse Precision, F1 and Kappa score. On the regression (

Table 2.10) task, DSCP outperforms DeepSynergy, DTF, CoSTCo, CP-WOPT and PRODeepSyn on nearly all metrics. DSCP only performs worse than Average baseline.

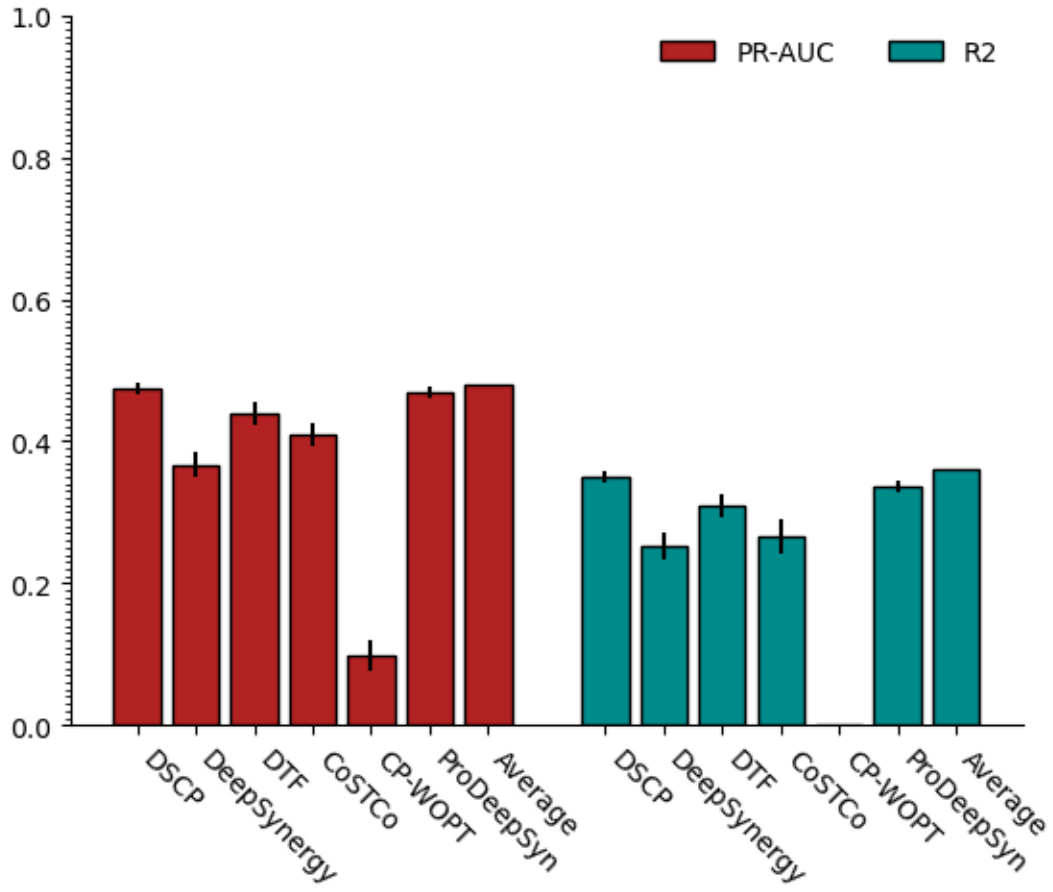

Figure 2.10 A summary of the classification and regression results on the Merick and Co compound screen using LCLO cross-validation.

Table 2.9 The classification results of our compared algorithms on the Merick and Co compound screen using LCLO cross-validation. *Red indicates that the algorithm performed worse than DSCP, green indicates better and black indicates that the result was insignificant. The statistics test used is outlined in section 2.3.6.*

Algorithm	ROC-AUC	PR-AUC	Recall	Prec.	F1	BACC	Kappa
Average	0.86 ± 0.00	0.48 ± 0.00	0.73 ± 0.00	0.31 ± 0.00	0.43 ± 0.00	0.78 ± 0.00	0.35 ± 0.00
DeepSynergy	0.85 ± 0.01	0.38 ± 0.01	0.64 ± 0.05	0.33 ± 0.02	0.43 ± 0.01	0.76 ± 0.01	0.36 ± 0.02
DTF	0.84 ± 0.02	0.44 ± 0.02	0.62 ± 0.03	0.32 ± 0.02	0.42 ± 0.02	0.74 ± 0.01	0.35 ± 0.02
CoSTCo	0.82 ± 0.01	0.41 ± 0.02	0.57 ± 0.10	0.32 ± 0.05	0.40 ± 0.02	0.72 ± 0.03	0.33 ± 0.03
CP-WOPT	0.49 ± 0.03	0.10 ± 0.02	0.15 ± 0.19	0.03 ± 0.04	0.06 ± 0.07	0.50 ± 0.01	0.00 ± 0.01
PRODeepSyn	0.86 ± 0.00	0.47 ± 0.01	0.64 ± 0.05	0.35 ± 0.03	0.45 ± 0.01	0.76 ± 0.02	0.38 ± 0.02
DSCP	0.86 ± 0.01	0.47 ± 0.01	0.80 ± 0.02	0.25 ± 0.02	0.38 ± 0.02	0.78 ± 0.01	0.29 ± 0.02

Table 2.10 The regression results of our compared algorithms on the Merick and Co compound screen using LCLO cross-validation. *Red indicates that the algorithm performed worse than DSCP, green indicates better and black indicates that the result was insignificant. The statistics test used is outlined in section 2.3.6.*

Algorithm	R^2	RMSE	MAE	MDAE
Average	0.36 ± 0.00	16.78 ± 0.00	10.97 ± 0.00	7.43 ± 0.00
DeepSynergy	0.27 ± 0.01	17.98 ± 0.10	11.68 ± 0.09	7.94 ± 0.11
DTF	0.31 ± 0.02	17.44 ± 0.21	11.66 ± 0.15	8.21 ± 0.19
CoSTCo	0.26 ± 0.02	17.98 ± 0.30	12.10 ± 0.32	8.50 ± 0.30
CP-WOPT	0.00 ± 0.00	21.00 ± 0.03	14.45 ± 0.03	10.22 ± 0.06
PRODeepSyn	0.34 ± 0.01	17.10 ± 0.10	11.33 ± 0.11	7.76 ± 0.14
DSCP	0.35 ± 0.01	16.93 ± 0.10	11.15 ± 0.09	7.72 ± 0.08

2.4.3 NCI-ALMANAC Compound Screen

2.4.3.1 Leave-Drug-Combination-Out (LDCO)

Table 2.11 The results of the hyperparameter search for DSCP on the NCI-ALMANAC compound screen using the LDCO cross-validation strategy.

Hyperparameter	Considered Values	Chosen Value	Importance
Rank	1-256	206	17%
MLP Architecture	[1024 1024 512], [2048 2048 1024], [2048 1024 512], [2048 2048 2048], [1024 1024 1024], [2048 2048], [1024 1024], [512 512], [1024 512]	[2048 2048 2048]	14%
Hidden Layers DR	0-0.5	0.3	7%
Input Layers DR	0-0.2	0.2	59%
Batch Size	1024, 2048,4096	1024	2%

From the hyperparameter search (**Table 2.11**) we can see that the input layer DR was found to be the most important for the prediction of DSCP's cross-validation performance. The remaining hyperparameters received a fairly even spread of importance. The MLP architecture was selected to be much larger than any of the architecture for the Merick and Co compound screen. This is sensible, as NCI-ALMANAC is significantly larger. On the classification (**Table 2.12**) task, DSCP performed worse than PRODeepSyn, DeepSynergy and DTF on most metrics. The comparison between DSCP, CoSTCo and CP-WOPT is mixed. On the regression (**Table 2.13**) task, DSCP performs worse the PRODeepSyn, DeepSynergy and DTF on most metrics and is consistently better then CoSTCo and CP-WOPT.

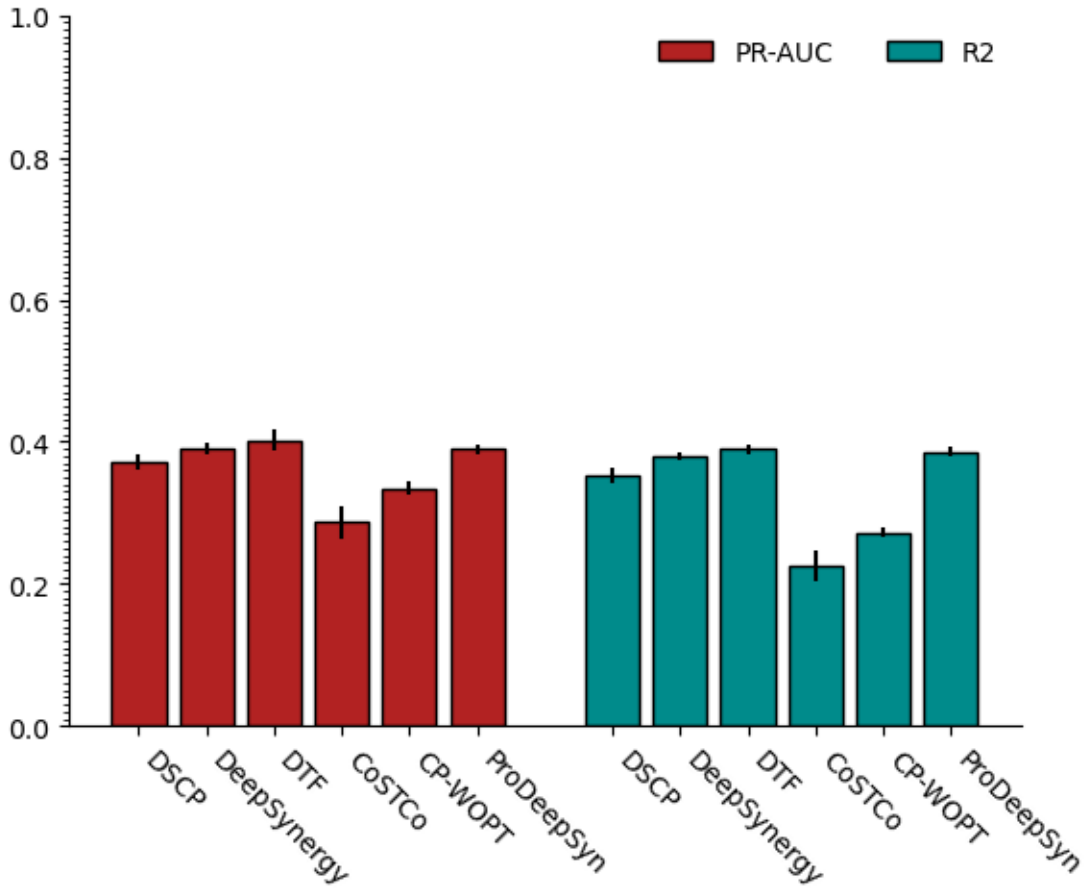

Figure 2.11 A summary of the classification and regression results on the NCI-ALMANAC compound screen using LDCO cross-validation.

Table 2.12 The classification results of our compared algorithms on the NCI-ALMANAC compound screen using LDCO cross-validation. *Red indicates that the algorithm performed worse than DSCP, green indicates better and black indicates that the result was insignificant. The statistics test used is outlined in section 2.3.6.*

Algorithm	ROC-AUC	PR-AUC	Recall	Prec.	F1	BACC	Kappa
DeepSynergy	0.85 ± 0.00	0.39 ± 0.01	0.40 ± 0.06	0.47 ± 0.06	0.42 ± 0.02	0.69 ± 0.03	0.41 ± 0.02
DTF	0.86 ± 0.01	0.40 ± 0.01	0.48 ± 0.07	0.41 ± 0.06	0.43 ± 0.01	0.73 ± 0.03	0.41 ± 0.01
CoSTCo	0.81 ± 0.01	0.29 ± 0.02	0.62 ± 0.02	0.15 ± 0.01	0.24 ± 0.01	0.75 ± 0.01	0.20 ± 0.01
CP-WOPT	0.83 ± 0.01	0.34 ± 0.01	0.58 ± 0.02	0.21 ± 0.01	0.31 ± 0.01	0.75 ± 0.01	0.27 ± 0.01
PRODeepSyn	0.85 ± 0.00	0.39 ± 0.01	0.41 ± 0.06	0.45 ± 0.06	0.42 ± 0.01	0.70 ± 0.03	0.40 ± 0.01
DSCP	0.84 ± 0.00	0.37 ± 0.01	0.35 ± 0.05	0.48 ± 0.05	0.40 ± 0.02	0.67 ± 0.03	0.38 ± 0.02

Table 2.13 The regression results of our compared algorithms on the NCI-ALMANAC compound screen using LDCO cross-validation. *Red indicates that the algorithm performed worse than DSCP, green indicates better and black indicates that the result was insignificant. The statistics test used is outlined in section 2.3.6.*

Algorithm	R^2	RMSE	MAE	MDAE
DeepSynergy	0.38 ± 0.01	5.77 ± 0.03	3.77 ± 0.02	2.61 ± 0.02
DTF	0.39 ± 0.01	5.73 ± 0.03	3.80 ± 0.01	2.67 ± 0.01
CoSTCo	0.23 ± 0.02	6.45 ± 0.09	4.36 ± 0.07	3.14 ± 0.07
CP-WOPT	0.27 ± 0.01	6.25 ± 0.03	4.19 ± 0.01	2.97 ± 0.01
PRODeepSyn	0.39 ± 0.01	4.74 ± 0.04	3.77 ± 0.02	2.62 ± 0.01
DSCP	0.35 ± 0.01	5.90 ± 0.05	3.84 ± 0.02	2.65 ± 0.02

2.5 Discussion

DSCP generally outperforms DTF at predicting the labels of the synthetic tensor when given the same rank. This is expected as DSCP has the ability to filter out irrelevant information while DTF does not. DSCP generally outperforms DTF when the synthetic tensor has varying levels of missing data, showing that it is robust to the presence of missing data. DSCP generally

outperforms DTF when the tensor's labels have severe class imbalance. In summary, DSCP outperforms DTF in a variety of common and extreme settings.

DSCP is selected to have a rank that is on the higher end of the hyperparameter search range. One of the primary benefits that we anticipated DSCP would have, is that it could achieve higher performances with a significantly lower rank than DTF by selectively choosing only the most important information. We observed this effect on the synthetic tensor, but the fact that DSCP preferred a higher rank on the ACDSP task implies it was not observed here.

DSCP appears to perform worse than DTF, PRODeepSyn and DeepSynergy on most metrics and datasets. DSCP appears to perform better than CP-WOPT on most metrics and datasets. Finally, the comparison between DSCP and CoSTCo was quite mixed and insignificant. The Merick-LCLO case is an exception to these observations, where DSCP outperformed all other algorithms, except the Average baseline, on most metrics. Additionally, on this dataset, DSCP chose a low rank (33).

2.6 Summary

In conclusion, we proposed a DTC algorithm, named DSCP, that is trainable in an end-to-end fashion. We show that the addition of end-to-end training results in an increase in the ability of DSCP to make inferences about a synthetic tensor. When applying DSCP to real-world tensors resulting from anticancer drug synergy compound screens, we do not observe an increase in performance. Though it appears that it has limited use on the ACDSP problem, DSCP may have more utility on real-world problems that have conditions which more closely match the synthetic data generation process used in this chapter. Finally, we show that SOTA ACDSP

algorithms fail to outperform a simple averaging baseline on the LDO and LCLO variants of the ACDSP problem.

Chapter 3 Artificial Intelligence-Driven Multimodal Data Fusion Techniques for Deep Tensor Completion-Based Anticancer Drug Synergy Prediction

3.1 Introduction

DTC algorithms such as DTF and DSCP have been shown to be able to achieve SOTA performance on the ACDSP task without the use of physicochemical or multiomic features related to the drugs or cell lines. We believe that DTC algorithms learn to make predictions using a style of reasoning distinct from that used than traditional ACDSP algorithms. This is due to the fact that the two classes of algorithms use entirely different sources of data to generate features for the drugs and cell lines. We hypothesize that an ACDSP algorithm that makes effective use of both tensor-based and physicochemical/multiomic-based features will be able to leverage both styles of reasonings and make better predictions than any one alone.

For each drug and cell line we are interested in making a prediction about, we have two modalities of data. For drugs, we have tensor-based features as well as physicochemical features. For cell lines, we have tensor-based features as well as genomic features. It is possible to combine these two modalities by simply concatenating the feature vectors for each drug and cell line before they are input into the algorithm. Though this is simple, there is evidence that for biomedical applications Intermediate Fusion (IF) [45] is better able to combine two complex data modalities. Specifically, we choose to embed both feature vectors into a low dimensional latent space, and then concatenate them. This has the advantage that it allows the algorithm to remove

information in each modality that is not relevant for the task at hand before combining distinct modalities.

Using SCP to extract tensor-based features, we compress both modalities using encoder-style MLPs. Following the recommendation of Stahlschmidt et al. [45], after combining the two data modalities, we reconstruct both the drug synergy tensor as well as the physicochemical and multiomic feature matrices using decoder-style MLPs. We name the resulting algorithm SynergyIF. We performed an extensive analysis comparing SynergyIF to six ACDSP algorithms on two separate compound screens. Additionally, we explore the utility of SynergyIF on the LDO and LCLO variants of the ACDSP task.

3.2 Materials and Methods

We re-conceptualize the problem of ACDSP as the compression, fusion, and reconstruction of three distinct data sources: past drug combination synergy data—in the form of a tensor $\mathcal{X}$--, drug feature matrix $\boldsymbol{D}$ and cell line feature matrix $\boldsymbol{CL}$. For the drug and cell line feature matrices, we utilize the physicochemical and genetic properties outlined in section 2.3.1.2.

First, our goal is to compress each source of data down to a low dimensional embedding. For the drug synergy tensor, we first apply SCP to extract drug and cell line specific features.

$$\boldsymbol{C}^{(d)}, \boldsymbol{C}^{(c)} = SCP(\mathcal{X}, \mathcal{W}; \boldsymbol{P}^{(d)}, \boldsymbol{P}^{(c)}) \quad (3.1)$$

We then use encoder-style MLPs on both the tensor features and the auxiliary data features.

$$\boldsymbol{V}^{(d)} = MLP\left(\boldsymbol{C}^{(d)}; \mathbb{W}_{Drug\ Tensor\ Embed}\right) \quad (3.2)$$

$$\boldsymbol{V}^{(c)} = MLP\left(\boldsymbol{C}^{(c)}; \mathbb{W}_{Cell\ Line\ Tensor\ Embed}\right)$$

(3.3)

$$\boldsymbol{Z}^{(d)} = MLP\left(\boldsymbol{D}; \mathbb{W}_{Drug\ Auxillary\ Embed}\right)$$

(3.4)

$$\boldsymbol{Z}^{(c)} = MLP\left(\boldsymbol{CL}; \mathbb{W}_{Cell\ Line\ Auxillary\ Embed}\right)$$

(3.5)

Then, we fuse together the two embeddings for the drugs and the two embeddings for the cell lines.

$$\boldsymbol{U} = [\boldsymbol{V}|\boldsymbol{Z}]$$

(3.6)

Finally, using the fused embeddings and decoder-style MLPs, we reconstruct the original synergy tensor, with missing values filled in, cell line feature matrix, and drug feature matrix.

$$\widetilde{\mathcal{X}} = MLP(\boldsymbol{U}^{(d)}, \boldsymbol{U}^{(c)}; \mathbb{W}_{Tensor\ Reconstruct})$$

(3.7)

$$\widetilde{\boldsymbol{D}} = MLP(\boldsymbol{U}^{(d)}; \mathbb{W}_{Drug\ Reconstruct})$$

(3.8)

$$\widetilde{\boldsymbol{CL}} = MLP(\boldsymbol{U}^{(c)}; \mathbb{W}_{Cell\ Line\ Reconstruct})$$

(3.9)

Our goal is to find the parameters to this algorithm that minimizes the reconstruction loss of the drug synergy tensor, drug auxiliary matrix and cell line auxiliary matrix.

$$\mathcal{L}_{Tensor\ Loss} = \left\| \mathcal{W} * (\mathcal{X} - \widetilde{\mathcal{X}})^2 \right\|$$

(3.10)

$$\mathcal{L}_{Drug\ Loss} = \left\| (\boldsymbol{D} - \widetilde{\boldsymbol{D}})^2 \right\|$$

(3.11)

$$\mathcal{L}_{Cell\ Line\ Loss} = \left\| (\boldsymbol{CL} - \widetilde{\boldsymbol{CL}})^2 \right\|$$

(3.12)

We apply a tunable weight to the drug and cell line reconstruction losses to allow us to adjust the importance of the drug synergy tensor reconstruction loss relative to the auxiliary data reconstruction losses.

$$\mathcal{L} = \mathcal{L}_{Tensor\ Loss} + \lambda \times \mathcal{L}_{Drug\ Loss} + \lambda \times \mathcal{L}_{Cell\ Line\ Loss} \quad (3.13)$$

This algorithm can be optimized using any stochastic gradient descent algorithm [50]. We depict the complete algorithm in **Figure 3.2** with legend **Figure 3.1**.

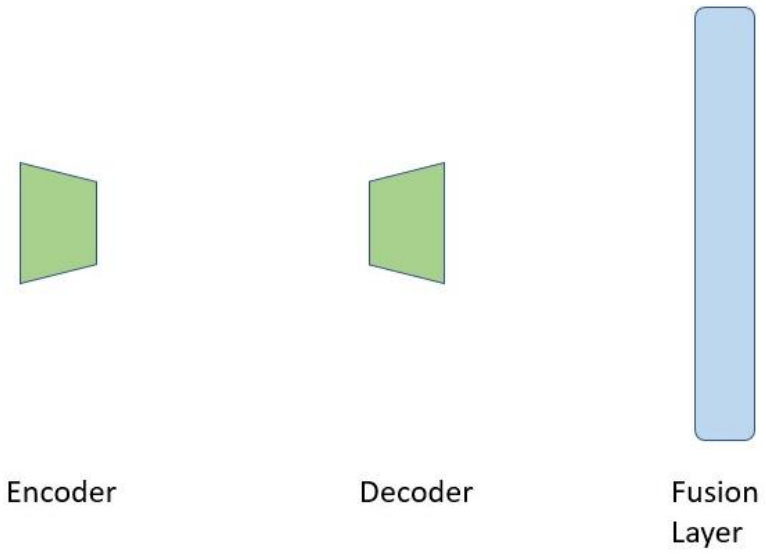

Figure 3.1 A legend for SynergyIF

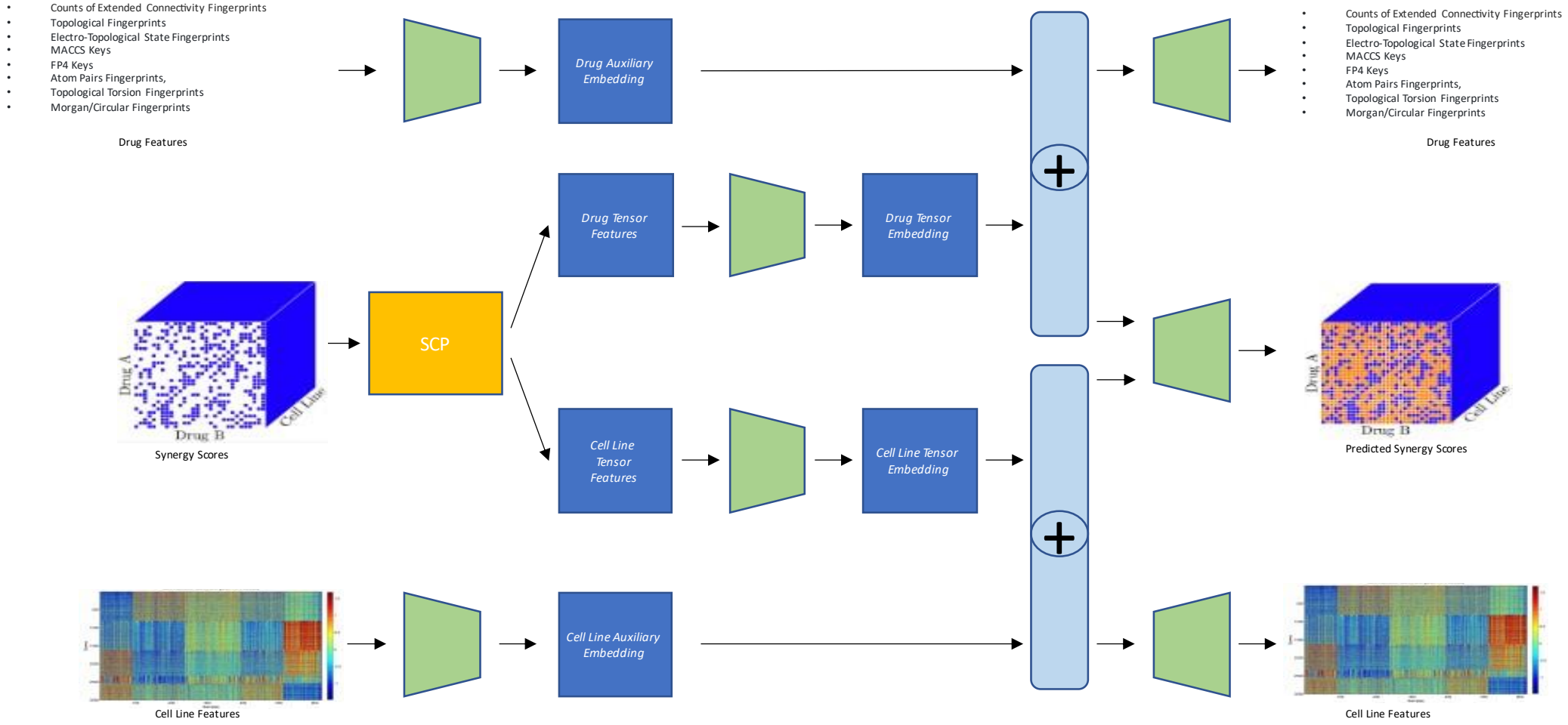

Figure 3.2 A depiction of SynergyIF.

3.2.1 Hyperparameter Description

What follows is a description of SynergyIF's hyperparameters.

- Rank: The number of columns in both the parameter and factor matrices. This hyperparameter influences the expressivity of SynergyIF, with higher values allowing for more expressivity and lower values having a regularization effect.
- Tensor Decoder MLP Architecture: The size of each layer in the tensor decoder MLP.
- Encoder MLP Architecture: The size of each layer in all encoder MLP's of SynergyIF.
- Auxiliary Decoder MLP Architecture: The size of each layer in the auxiliary decoder MLP's of SynergyIF. This is set to be the exact reverse architecture as the encoders, and thus is not included in any hyperparameter search.
- Hidden Layers DR: The dropout rate of each hidden layer of the MLP. Each of the MLP's described above have different Hidden Layer DR's.
- Input Layers DR: The dropout rate of the output of the first layer of the MLP. Each of the MLP's described above have different input layer DR's.
- Auxiliary Weight:
- Embedding Dimension: The dimension of both the tensor and auxiliary encoders. This controls the expressivity of SynergyIF in a manner similar to the rank.
- Batch Size: The batch size used in the stochastic gradient descent algorithm.

3.3 Data Sources and Algorithm Evaluation Strategy

Table 3.1 A summary of the algorithms, data sources and cross-validation strategies considered in this chapter.

Algorithms
DTF [4], DeepSynergy[35], CoSTCo [18], CP-WOPT [13], PRODeepSyn [37], Average, DSCP, SynergyIF
Data Sources
Merick and Co [2], NCI-ALMANAC [47]
Cross-Validation Strategies
Leave-Drug-Combination-Out (LDCO), Leave-Cell-Line-Out (LCLO), Leave-Drug-Out (LDO)
Metrics
ROC-AUC, PR-AUC, Precision, Recall, F1-Score, BACC, Kappa, R2, MAE, MDAE, RMSE
Software
TensorFlow (2.4.1)[51], Tensorly (0.6.0) [63], SciPy (1.7.3) [64], Scikit-Learn (0.24.2) [58]

The data sources and evaluation strategy for this chapter stays largely the same as discussed in section 2.3. The compared algorithms, data sources and cross-validation strategies used are summarized in **Table 3.1**. We utilize the same metrics, hyperparameter optimization and test set inference strategy, as discussed in section 2.3.

3.4 Results

3.4.1 Merick and Co Compound Screen

3.4.1.1 Leave-Drug-Combination-Out (LDCO)

Table 3.2 The results of the hyperparameter search for SynergyIF on the Merick and Co compound screen using the LDCO cross-validation strategy.

Hyperparameter	Considered Values	Chosen Value	Importance
Rank	1-512	406	2%
Tensor Decoder MLP Architecture	[1024 1024 512], [2048 2048 1024], [2048 1024 512], [2048 2048 2048], [1024 1024 1024], [2048 2048], [1024 1024], [512 512], [1024 512]	[1024 1024]	8%
Tensor Decoder Hidden Layers DR	0-0.5	0.2	0%
Tensor Decoder Input Layers DR	0-0.2	0.1	0%
Encoder MLP Architecture	[2048 1024 512], [1024 512 256], [512 256 128], [1024 512], [512 256], [256 128]	[512 256 128]	2%
Encoder Hidden Layers DR	0-0.5	0.0	16%
Encoder Input Layers DR	0-0.2	0.2	1%
Auxiliary Decoder Hidden Layers DR	0-0.5	0.5	3%
Auxiliary Decoder Input Layers DR	0-0.2	0.1	1%
Auxiliary Weight	0-10	3.0631	1%
Embedding Dimension	1-128	117	62%
Batch Size	512, 1024, 2048	512	4%

From the hyperparameter search (**Table 3.2**) we can see that the parameters that best predict the performance of SynergyIF are the embedding dimension (62%), the hidden layer DR in the encoders (16%) and the tensor decoder architecture (8%). Interestingly, the weight on the auxiliary loss was not predictive of SynergyIF's performance (1%). This suggests that SynergyIF is indifferent to whether the auxiliary data is reconstructed. The embedding dimension was by far the most predictive of the performance of SynergyIF and was selected to be 117. For the embedding to be seen as a highly compressed version of the drug and cell line features, its dimension would need to be significantly less than the number of drugs (39) and cell lines (38) in the dataset. This suggests that compressing the data sources to a low-dimensional embedding reduces the performance of SynergyIF.

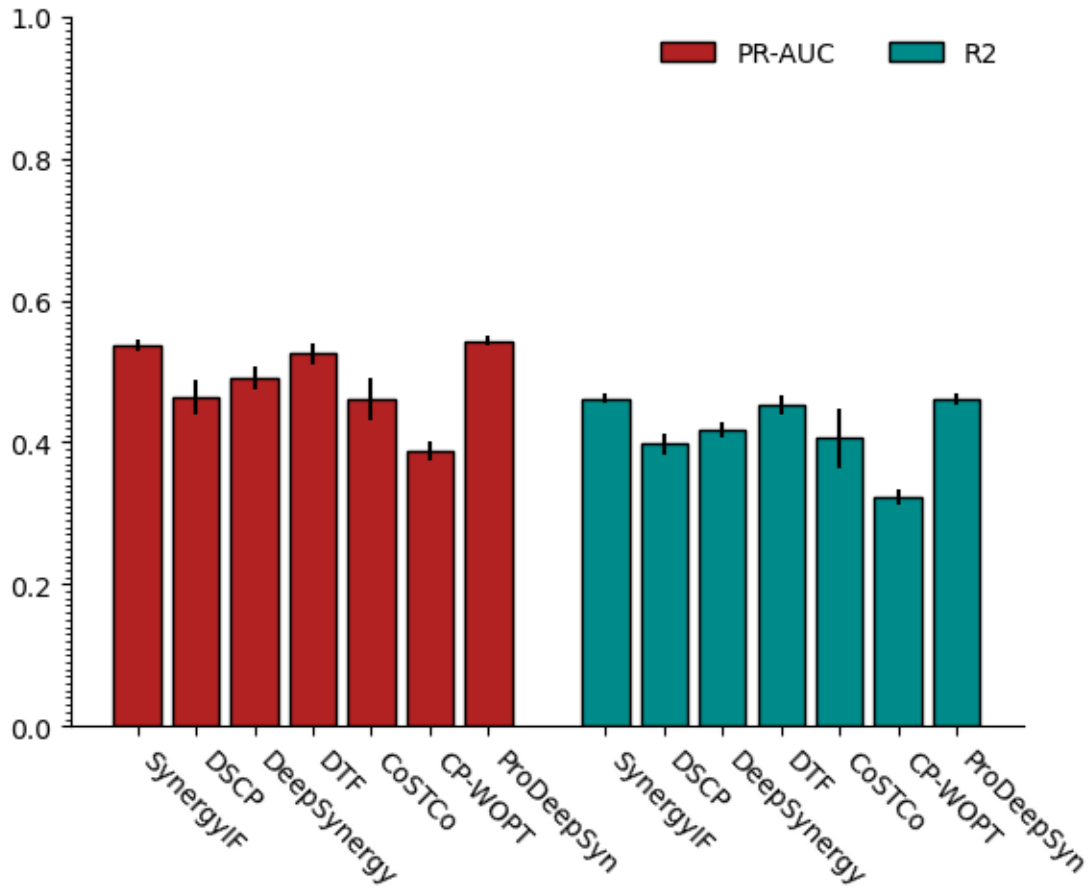

Figure 3.3 A summary of the classification and regression results on the Merick and Co compound screen using LDCO cross-validation.

On the regression task (**Table 3.4**), SynergyIF achieves statistically significantly better results than DeepSynergy, CoSTCo, CP-WOPT and DSCP on all metrics. At the 95% confidence level, we were unable to disprove that SynergyIF performs the same as PRODeepSyn and DTF on most regression metrics. On the classification task (**Table 3.3**), SynergyIF's performance in relation to our baseline algorithms is largely inconclusive. SynergyIF has lower Recall and higher Precision than most other algorithms, showing that it makes more conservative

predictions. For every classification metric, we were unable to disprove that SynergyIF and PRODeepSyn perform the same.

Table 3.3 The classification results of our compared algorithms on the Merick and Co compound screen using LDCO cross-validation. *Red indicates that the algorithm performed worse than DSCP, green indicates better and black indicates that the result was insignificant. The statistics test used is outlined in section 2.3.6.*

Algorithm	ROC-AUC	PR-AUC	Recall	Prec.	F1	BACC	Kappa
DeepSynergy	0.84 ± 0.01	0.50 ± 0.02	0.54 ± 0.07	0.44 ± 0.08	0.47 ± 0.02	0.74 ± 0.02	0.42 ± 0.03
DTF	0.84 ± 0.01	0.52 ± 0.01	0.49 ± 0.04	0.51 ± 0.07	0.49 ± 0.02	0.72 ± 0.01	0.45 ± 0.02
CoSTCo	0.81 ± 0.02	0.46 ± 0.03	0.50 ± 0.05	0.40 ± 0.03	0.45 ± 0.02	0.72 ± 0.02	0.39 ± 0.03
CP-WOPT	0.79 ± 0.01	0.39 ± 0.01	0.41 ± 0.04	0.38 ± 0.03	0.39 ± 0.02	0.68 ± 0.02	0.34 ± 0.02
PRODeepSyn	0.85 ± 0.00	0.54 ± 0.01	0.42 ± 0.04	0.62 ± 0.05	0.49 ± 0.01	0.70 ± 0.02	0.46 ± 0.01
DSCP	0.82 ± 0.01	0.46 ± 0.02	0.50 ± 0.03	0.41 ± 0.04	0.45 ± 0.03	0.72 ± 0.01	0.39 ± 0.03
SynergyIF	0.85 ± 0.00	0.54 ± 0.01	0.40 ± 0.01	0.63 ± 0.02	0.49 ± 0.01	0.69 ± 0.00	0.46 ± 0.01

Table 3.4 The regression results of our compared algorithms on the Merick and Co compound screen using LDCO cross-validation. *Red indicates that the algorithm performed worse than DSCP, green indicates better and black indicates that the result was insignificant. The statistics test used is outlined in section 2.3.6.*

Algorithm	R^2	RMSE	MAE	MDAE
DeepSynergy	0.42 ± 0.02	16.86 ± 0.32	10.40 ± 0.23	6.93 ± 0.20
DTF	0.45 ± 0.01	16.45 ± 0.20	10.22 ± 0.19	6.78 ± 0.17
CoSTCo	0.41 ± 0.04	17.12 ± 0.60	11.28 ± 0.39	7.83 ± 0.29
CP-WOPT	0.32 ± 0.01	18.29 ± 0.15	11.97 ± 0.17	8.40 ± 0.16
PRODeepSyn	0.46 ± 0.01	16.32 ± 0.10	10.05 ± 0.10	6.70 ± 0.13
DSCP	0.40 ± 0.01	17.24 ± 0.21	10.93 ± 0.17	7.40 ± 0.15
SynergyIF	0.46 ± 0.01	16.30 ± 0.10	10.05 ± 0.06	6.65 ± 0.09

3.4.1.2 Leave-Drug-Out (LDO)

Table 3.5 The results of the hyperparameter search for SynergyIF on the Merick and Co compound screen using the LDO cross-validation strategy.

Hyperparameter	Considered Values	Chosen Value	Importance
Rank	1-512	248	4%
Tensor Decoder MLP Architecture	[1024 1024 512], [2048 2048 1024], [2048 1024 512], [2048 2048 2048], [1024 1024 1024], [2048 2048], [1024 1024], [512 512], [1024 512]	[1024 1024]	4%
Tensor Decoder Hidden Layers DR	0-0.5	0.4	1%
Tensor Decoder Input Layers DR	0-0.2	0.2	1%
Encoder MLP Architecture	[2048 1024 512], [1024 512 256], [512 256 128], [1024 512], [512 256], [256 128]	[2048 1024 512]	6%
Encoder Hidden Layers DR	0-0.5	0.0	65%
Encoder Input Layers DR	0-0.2	0.2	1%
Auxiliary Decoder Hidden Layers DR	0-0.5	0.3	3%
Auxiliary Decoder Input Layers DR	0-0.2	0.1	7%
Auxiliary Weight	0-10	0.2609	2%
Embedding Dimension	1-128	62	6%
Batch Size	512, 1024, 2048	1024	0%

From the hyperparameter search (**Table 3.5**) we can see that the most important hyperparameters were related to the expressivity of the encoder, with the encoder hidden layer DR at 65%, the encoder input layer DR at 7% and the encoder architecture at 6%. These hyperparameters were selected such that the expressivity of the encoder would be very high with low DR and a large MLP architecture. The next most important hyperparameter was the embedding dimension at 6% with a dimension significantly higher than the number of drugs and cell lines in the compound screen being chosen. The weight on the auxiliary loss was found to be unimportant at 2%, once again implying that the reconstruction of the auxiliary data was not found to be critical to SynergyIF's performance.

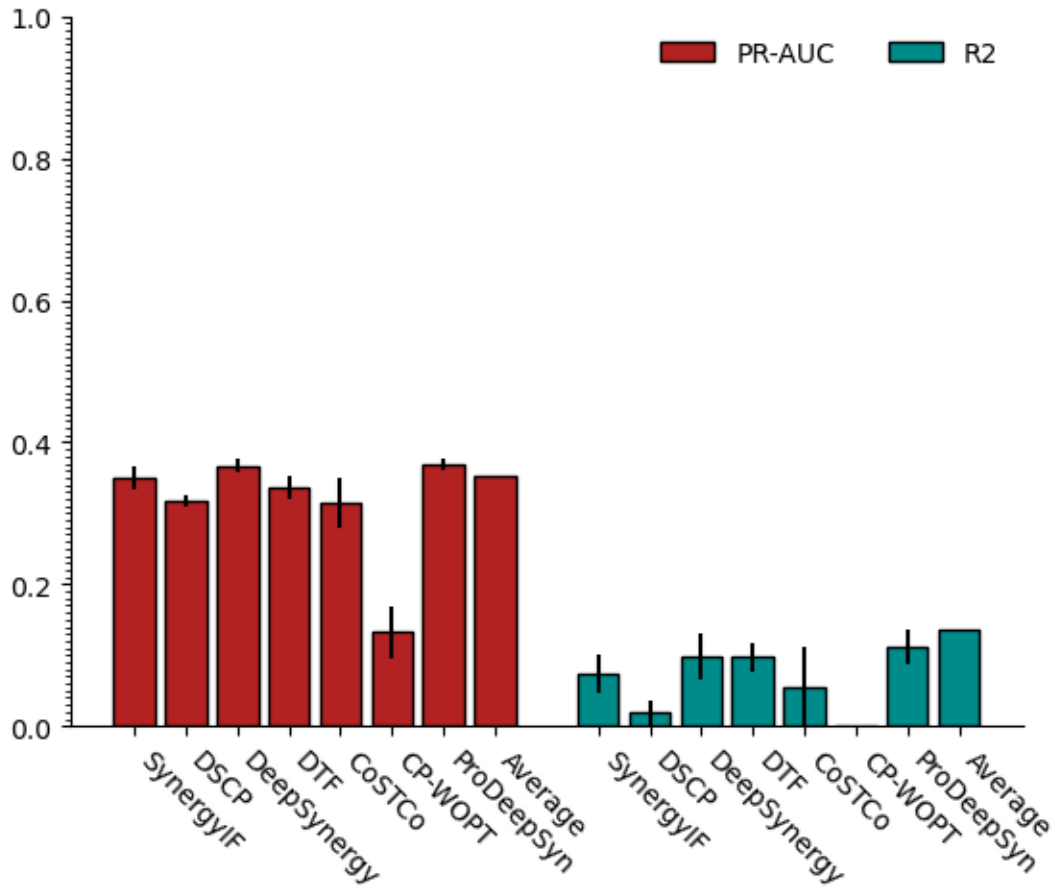

Figure 3.4 A summary of the classification and regression results on the Merick and Co compound screen using LDO cross-validation.

For most regression and classification metrics (**Table 3.6**, **Table 3.7**) SynergyIF performs significantly worse than both Average and PRODeepSyn. For most regression and classification metrics, it performs significantly better than DSCP and CP-WOPT. For the remaining algorithms, the comparison yields mixed results. Clearly SynergyIF fails to make high-quality predictions, performing worse than even the Average algorithm. Due to the high expressivity of the encoder and the large embedding dimension chosen by the hyperparameter search, SynergyIF

struggles with overfitting. On the training set, it achieves an R2 of 0.71 ± 0.05 compared to its test set performance of 0.07 ± 0.03.

Table 3.6 The classification results of our compared algorithms on the Merick and Co compound screen using LDO cross-validation. *Red indicates that the algorithm performed worse than DSCP, green indicates better and black indicates that the result was insignificant. The statistics test used is outlined in section 2.3.6.*

Algorithm	ROC-AUC	PR-AUC	Recall	Prec.	F1	BACC	Kappa
Average	0.74 ± 0.00	0.35 ± 0.00	0.32 ± 0.00	0.41 ± 0.00	0.36 ± 0.00	0.63 ± 0.00	0.28 ± 0.00
DeepSynergy	0.74 ± 0.01	0.37 ± 0.01	0.30 ± 0.04	0.46 ± 0.03	0.36 ± 0.02	0.63 ± 0.01	0.30 ± 0.02
DTF	0.72 ± 0.01	0.34 ± 0.02	0.20 ± 0.05	0.48 ± 0.05	0.28 ± 0.04	0.59 ± 0.02	0.23 ± 0.03
CoSTCo	0.70 ± 0.03	0.31 ± 0.04	0.31 ± 0.07	0.38 ± 0.05	0.33 ± 0.05	0.62 ± 0.03	0.26 ± 0.04
CP-WOPT	0.51 ± 0.05	0.13 ± 0.04	0.25 ± 0.25	0.08 ± 0.05	0.11 ± 0.08	0.50 ± 0.01	0.00 ± 0.01
PRODeepSyn	0.74 ± 0.00	0.37 ± 0.01	0.27 ± 0.03	0.48 ± 0.02	0.35 ± 0.02	0.62 ± 0.01	0.29 ± 0.02
DSCP	0.69 ± 0.01	0.32 ± 0.01	0.12 ± 0.01	0.58 ± 0.03	0.20 ± 0.02	0.56 ± 0.01	0.17 ± 0.01
SynergyIF	0.72 ± 0.01	0.35 ± 0.02	0.20 ± 0.04	0.53 ± 0.02	0.28 ± 0.04	0.59 ± 0.02	0.23 ± 0.03

Table 3.7 The regression results of our compared algorithms on the Merick and Co compound screen using LDO cross-validation. *Red indicates that the algorithm performed worse than DSCP, green indicates better and black indicates that the result was insignificant. The statistics test used is outlined in section 2.3.6.*

Algorithm	R^2	RMSE	MAE	MDAE
Average	0.14 ± 0.00	22.18 ± 0.00	14.41 ± 0.00	9.94 ± 0.00
DeepSynergy	0.10 ± 0.03	22.64 ± 0.41	14.82 ± 0.42	10.25 ± 0.40
DTF	0.10 ± 0.02	22.67 ± 0.26	14.75 ± 0.22	10.01 ± 0.17
CoSTCo	0.05 ± 0.06	23.18 ± 0.68	15.41 ± 0.54	10.74 ± 0.47
CP-WOPT	-0.07 ± 0.00	24.65 ± 0.02	16.04 ± 0.01	10.89 ± 0.02
PRODeepSyn	0.11 ± 0.02	22.48 ± 0.30	14.72 ± 0.32	10.19 ± 0.31
DSCP	0.02 ± 0.02	23.63 ± 0.22	15.66 ± 0.22	10.79 ± 0.23
SynergyIF	0.07 ± 0.03	22.95 ± 0.35	15.05 ± 0.31	10.30 ± 0.23

3.4.1.3 Leave-Cell-Line-Out (LCLO)

Table 3.8 The results of the hyperparameter search for SynergyIF on the Merick and Co compound screen using the LCLO cross-validation strategy.

Hyperparameter	Considered Values	Chosen Value	Importance
Rank	1-512	241	12%
Tensor Decoder MLP Architecture	[1024 1024 512], [2048 2048 1024], [2048 1024 512], [2048 2048 2048], [1024 1024 1024], [2048 2048], [1024 1024], [512 512], [1024 512]	[1024 512]	16%
Tensor Decoder Hidden Layers DR	0-0.5	0.5	8%
Tensor Decoder Input Layers DR	0-0.2	0.1	0%
Encoder MLP Architecture	[2048 1024 512], [1024 512 256], [512 256 128], [1024 512], [512 256], [256 128]	[1024 512]	7%
Encoder Hidden Layers DR	0-0.5	0.4	11%
Encoder Input Layers DR	0-0.2	0.1	15%
Auxiliary Decoder Hidden Layers DR	0-0.5	0.3	3%
Auxiliary Decoder Input Layers DR	0-0.2	0.2	0%
Auxiliary Weight	0-10	0.5057	7%
Embedding Dimension	1-128	119	14%
Batch Size	512, 1024, 2048	512	6%

The importance of each hyperparameter (**Table 3.8**) is distributed quite evenly, with an emphasis being put on the parameters that can control for overfitting. We hypothesize that this was because SynergyIF learned to predict the average and thus needed to have low expressivity to prevent it from deviating significantly from this. We examine this hypothesis by subtracting the average of each cell line in the training set from combinations in the testing set for both the ground truth and SynergyIF's predictions. We average SynergyIF's ten predictions for each sample to reduce variations between training sessions. We then plot the ground truth and SynergyIF's predictions against each other and analyze the correlation (**Figure 3.5**). We observe an R2 of 0.3403, which is in line with the results from **Table 3.10**. Then, in **Figure 3.6** we observe a strong correlation (R2=0.8608) between SynergyIF's prediction and the average of the drug pairs across the cell lines in the training set. Finally, we subtract the drug pair averages from both the ground truth labels and SynergyIF's predictions. Little correlation (R2=0.045) exists, strongly implying that SynergyIF simply learned to predict the average.

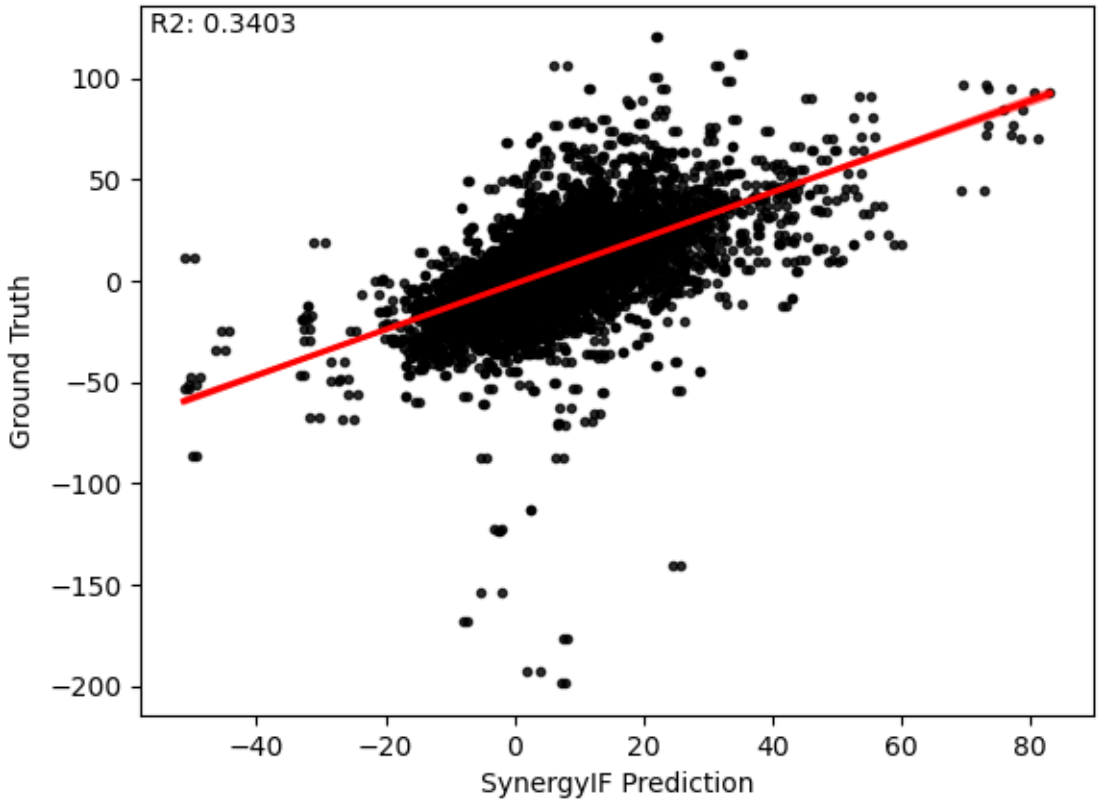

Figure 3.5 A scatter plot of SynergyIF's predictions against the ground truth.

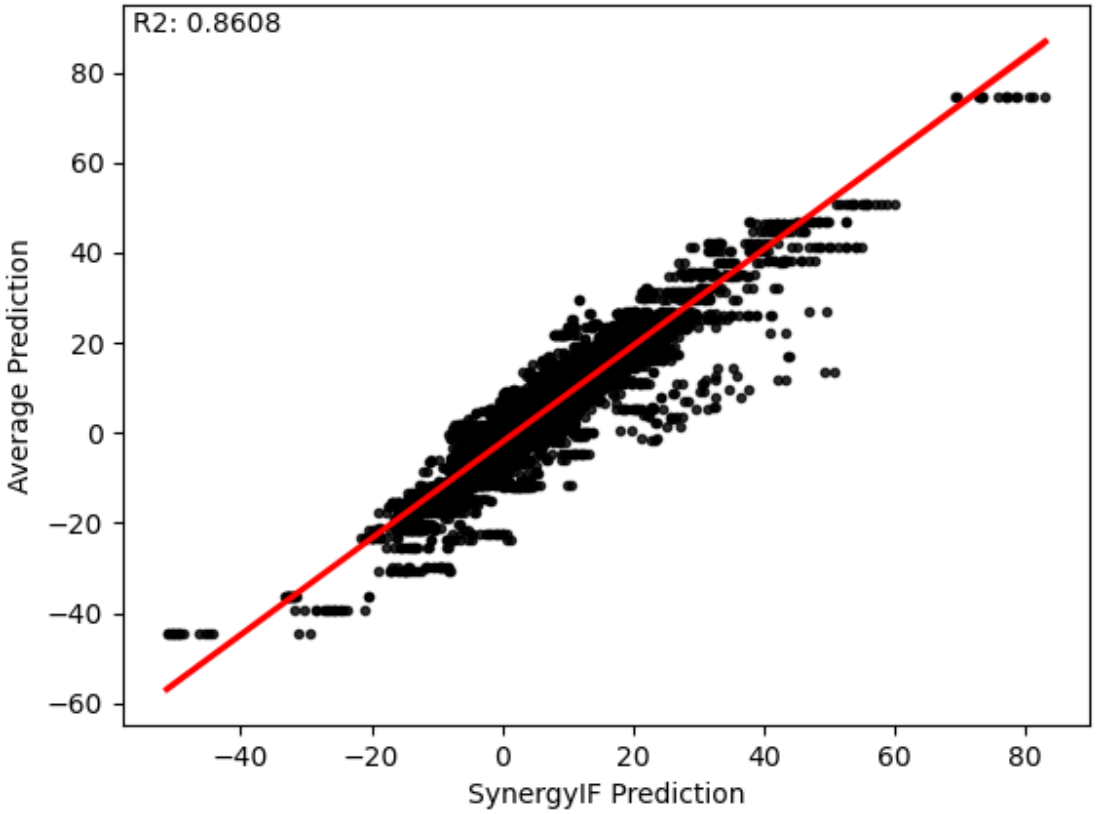

Figure 3.6 A scatter plot of SynergyIF's predictions against the average for each drug pair across every cell line in the training set.

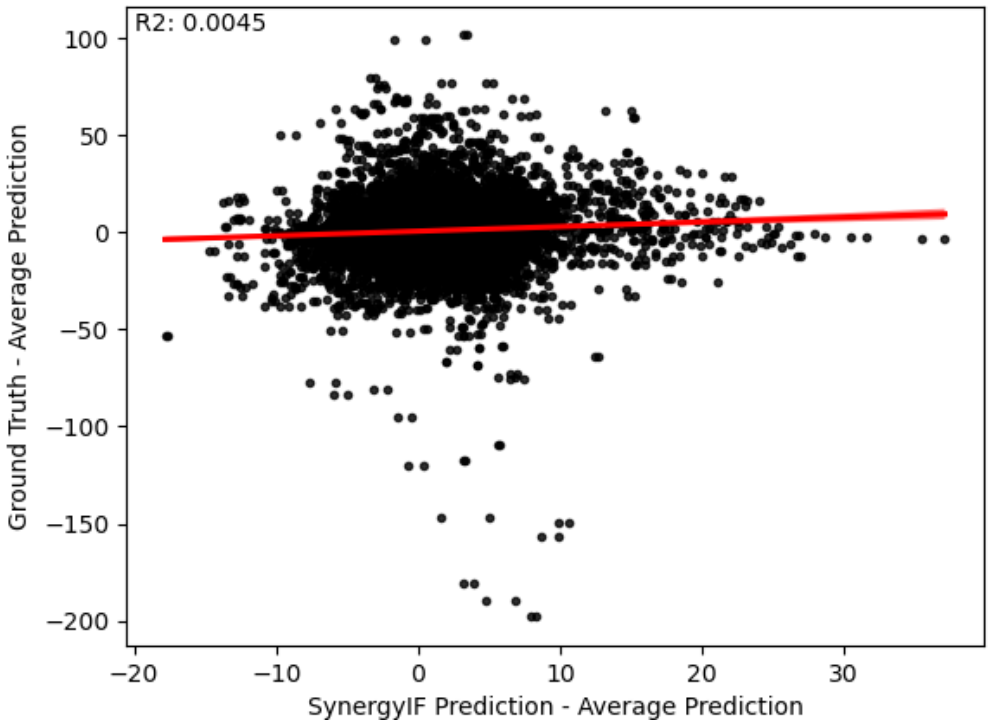

Figure 3.7 A scatter plot of SynergyIF's predictions against the ground truth with the average subtracted from both.

Across the board improvements in regression metrics (**Table 3.10**) by Average over SynergyIF suggests that the mixed results in the comparison between these algorithms on the classification task (**Table 3.9**) are due to differences in the threshold chosen. The fact that the two classification metrics that do not use a threshold (PR-AUC and ROC-AUC) show improvements of Average over SynergyIF supports this interpretation.

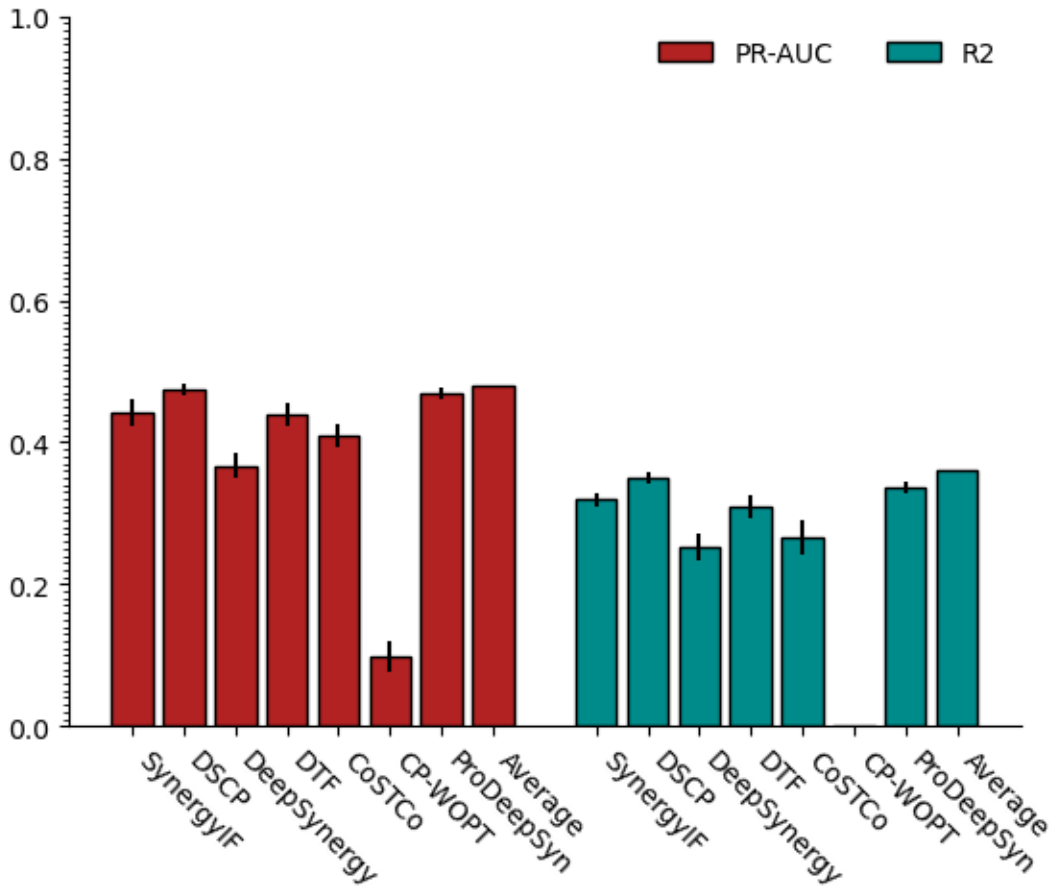

Figure 3.8 A summary of the classification and regression results on the Merick and Co compound screen using LCLO cross-validation.

Table 3.9 The classification results of our compared algorithms on the Merick and Co compound screen using LCLO cross-validation. *Red indicates that the algorithm performed worse than DSCP, green indicates better and black indicates that the result was insignificant. The statistics test used is outlined in section 2.3.6.*

Algorithm	ROC-AUC	PR-AUC	Recall	Prec.	F1	BACC	Kappa
Average	0.86 ± 0.00	0.48 ± 0.00	0.73 ± 0.00	0.31 ± 0.00	0.43 ± 0.00	0.78 ± 0.00	0.35 ± 0.00
DeepSynergy	0.85 ± 0.01	0.38 ± 0.01	0.64 ± 0.05	0.33 ± 0.02	0.43 ± 0.01	0.76 ± 0.01	0.36 ± 0.02
DTF	0.84 ± 0.02	0.44 ± 0.02	0.62 ± 0.03	0.32 ± 0.02	0.42 ± 0.02	0.74 ± 0.01	0.35 ± 0.02
CoSTCo	0.82 ± 0.01	0.41 ± 0.02	0.57 ± 0.10	0.32 ± 0.05	0.40 ± 0.02	0.72 ± 0.03	0.33 ± 0.03
CP-WOPT	0.49 ± 0.03	0.10 ± 0.02	0.15 ± 0.19	0.03 ± 0.04	0.06 ± 0.07	0.50 ± 0.01	0.00 ± 0.01
PRODeepSyn	0.86 ± 0.00	0.47 ± 0.01	0.64 ± 0.05	0.35 ± 0.03	0.45 ± 0.01	0.76 ± 0.02	0.38 ± 0.02
DSCP	0.86 ± 0.01	0.47 ± 0.01	0.80 ± 0.02	0.25 ± 0.02	0.38 ± 0.02	0.78 ± 0.01	0.29 ± 0.02
SynerygIF	0.85 ± 0.01	0.44 ± 0.02	0.59 ± 0.03	0.36 ± 0.02	0.44 ± 0.01	0.74 ± 0.01	0.37 ± 0.02

Table 3.10 The regression results of our compared algorithms on the Merick and Co compound screen using LCLO cross-validation. *Red indicates that the algorithm performed worse than DSCP, green indicates better and black indicates that the result was insignificant. The statistics test used is outlined in section 2.3.6.*

Algorithm	R^2	RMSE	MAE	MDAE
Average	0.36 ± 0.00	16.78 ± 0.00	10.97 ± 0.00	7.43 ± 0.00
DeepSynergy	0.27 ± 0.01	17.98 ± 0.10	11.68 ± 0.09	7.94 ± 0.11
DTF	0.31 ± 0.02	17.44 ± 0.21	11.66 ± 0.15	8.21 ± 0.19
CoSTCo	0.26 ± 0.02	17.98 ± 0.30	12.10 ± 0.32	8.50 ± 0.30
CP-WOPT	0.00 ± 0.00	21.00 ± 0.03	14.45 ± 0.03	10.22 ± 0.06
PRODeepSyn	0.34 ± 0.01	17.10 ± 0.10	11.33 ± 0.11	7.76 ± 0.14
DSCP	0.35 ± 0.01	16.93 ± 0.10	11.15 ± 0.09	7.72 ± 0.08
SynergyIF	0.32 ± 0.01	17.32 ± 0.12	11.40 ± 0.10	7.88 ± 0.10

3.4.2 NCI-ALMANAC Compound Screen

3.4.2.1 Leave-Drug-Combination-Out (LDCO)

Table 3.11 The results of the hyperparameter search for SynergyIF on the NCI-ALMANAC compound screen using the LDCO cross-validation strategy.

Hyperparameter	Considered Values	Chosen Value	Importance
Rank	1-512	330	1%
Tensor Decoder MLP Architecture	[1024 1024 512], [2048 2048 1024], [2048 1024 512], [2048 2048 2048], [1024 1024 1024], [2048 2048], [1024 1024], [512 512], [1024 512]	[2048 2048 2048]	4%
Tensor Decoder Hidden Layers DR	0-0.5	0.3	14%
Tensor Decoder Input Layers DR	0-0.2	0.1	0%
Encoder MLP Architecture	[2048 1024 512], [1024 512 256], [512 256 128], [1024 512], [512 256], [256 128]	[1024 512 256]	2%
Encoder Hidden Layers DR	0-0.5	0.2	0%
Encoder Input Layers DR	0-0.2	0.0	0%
Auxiliary Decoder Hidden Layers DR	0-0.5	0.2	4%
Auxiliary Decoder Input Layers DR	0-0.2	0.1	4%
Auxiliary Weight	0-10	1.2544	0%
Embedding Dimension	1-128	123	57%
Batch Size	512, 1024, 2048	512	0%

From the hyperparameter search table (**Table 3.11**) we can see that, as was the case with the Merick and Co compound screen, the embedding dimension is the hyperparameter that best predicts the performance of SynergyIF at 57% importance. Once again, an embedding dimension that is larger than the number of drugs or cell lines in the dataset was chosen. Unlike the Merick and Co compound screen, the next most important hyperparameters were the DR and MLP architecture of decoders. The weight on the auxiliary loss was found to not predict SynergyIF's performance at all, suggesting that the reconstruction of auxiliary data for this compound screen fails to improve SynergyIF's performance.

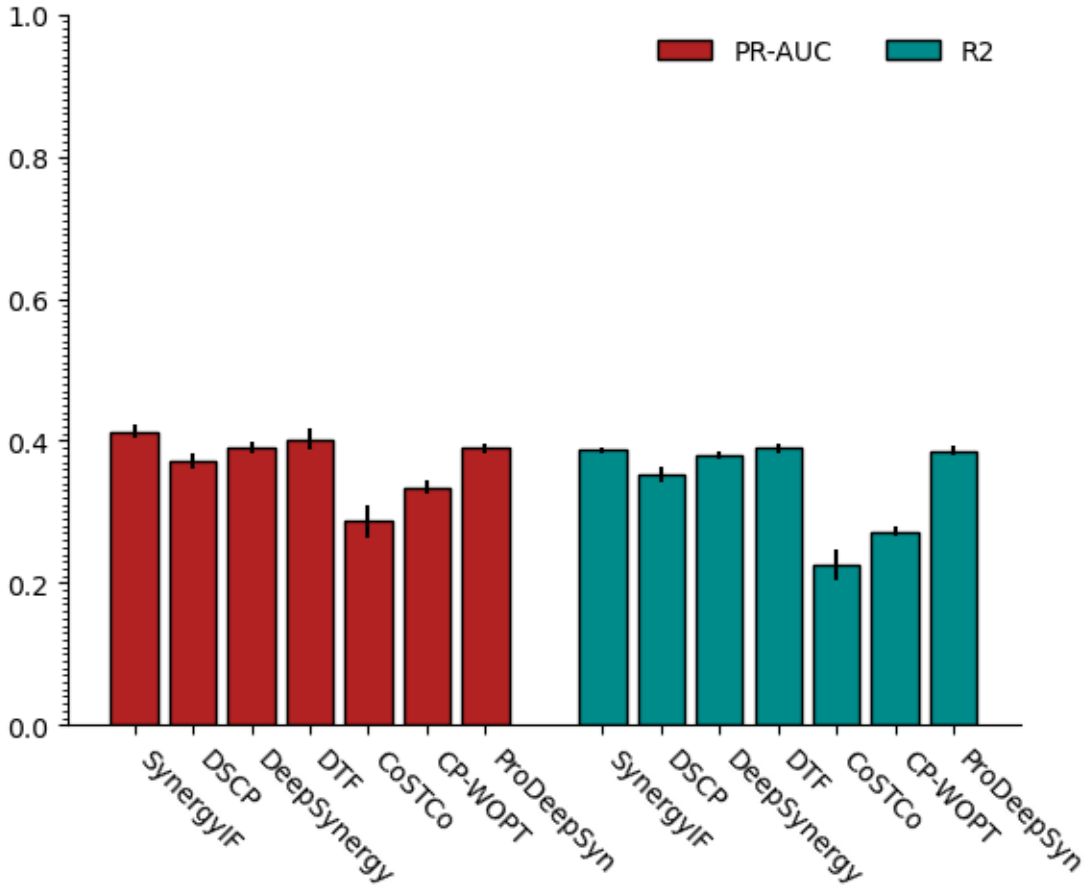

Figure 3.9 A summary of the classification and regression results on the NCI-ALMANAC compound screen using LDCO cross-validation.

Besides DTF, for which the comparison is inconclusive, SynergyIF outperforms every other algorithm on the PR-AUC and F1-Score metric and every other algorithm on the ROC-AUC metric (**Table 3.12**). The comparison is mixed for the remaining classification metrics. The comparison between SynergyIF and PRODeepSyn lacks evidence to reject the hypothesis that performance of these algorithms is identical for every regression metric (**Table 3.13**). SynergyIF significantly outperforms CoSTCo, CP-WOPT and DSCP on the regression task. When comparing SynergyIF, DeepSynergy and DTF, the results are mixed.

Table 3.12 The classification results of our compared algorithms on the NCI-ALMANAC compound screen using LDCO cross-validation. *Red indicates that the algorithm performed worse than DSCP, green indicates better and black indicates that the result was insignificant. The statistics test used is outlined in section 2.3.6.*

Algorithm	ROC-AUC	PR-AUC	Recall	Prec.	F1	BACC	Kappa
DeepSynergy	0.84 ± 0.01	0.37 ± 0.01	0.48 ± 0.03	0.36 ± 0.03	0.41 ± 0.01	0.72 ± 0.01	0.38 ± 0.01
DTF	0.86 ± 0.01	0.40 ± 0.01	0.48 ± 0.07	0.41 ± 0.06	0.43 ± 0.01	0.73 ± 0.03	0.41 ± 0.01
CoSTCo	0.81 ± 0.01	0.29 ± 0.02	0.62 ± 0.02	0.15 ± 0.01	0.24 ± 0.01	0.75 ± 0.01	0.20 ± 0.01
CP-WOPT	0.83 ± 0.01	0.34 ± 0.01	0.58 ± 0.02	0.21 ± 0.01	0.31 ± 0.01	0.75 ± 0.01	0.27 ± 0.01
PRODeepSyn	0.85 ± 0.00	0.39 ± 0.01	0.41 ± 0.06	0.45 ± 0.06	0.42 ± 0.01	0.70 ± 0.03	0.40 ± 0.01
DSCP	0.84 ± 0.00	0.37 ± 0.01	0.35 ± 0.05	0.48 ± 0.05	0.40 ± 0.02	0.67 ± 0.03	0.38 ± 0.02
SynergyIF	0.86 ± 0.01	0.42 ± 0.01	0.44 ± 0.06	0.46 ± 0.06	0.44 ± 0.01	0.71 ± 0.03	0.42 ± 0.01

Table 3.13 The regression results of our compared algorithms on the NCI-ALMANAC compound screen using LDCO cross-validation. *Red indicates that the algorithm performed worse than DSCP, green indicates better and black indicates that the result was insignificant. The statistics test used is outlined in section 2.3.6.*

Algorithm	R^2	RMSE	MAE	MDAE
DeepSynergy	0.36 ± 0.01	5.88 ± 0.04	3.86 ± 0.02	2.68 ± 0.01
DTF	0.39 ± 0.01	5.73 ± 0.03	3.80 ± 0.01	2.67 ± 0.01
CoSTCo	0.23 ± 0.02	6.45 ± 0.09	4.36 ± 0.07	3.14 ± 0.07
CP-WOPT	0.27 ± 0.01	6.25 ± 0.03	4.19 ± 0.01	2.97 ± 0.01
PRODeepSyn	0.39 ± 0.01	4.74 ± 0.04	3.77 ± 0.02	2.62 ± 0.01
DSCP	0.35 ± 0.01	5.90 ± 0.05	3.84 ± 0.02	2.65 ± 0.02
SynergyIF	0.39 ± 0.01	5.71 ± 0.03	3.74 ± 0.02	2.60 ± 0.02

3.5 Discussion

On the LDO and LCLO tasks, none of the SOTA algorithms perform better than the Average baseline algorithm. This shows that these tasks are extremely difficult and no SOTA algorithm has discovered a technique that allows them to make high-quality predictions. The LCLO task is significantly easier than the LDO task, as indicated by the Average baseline

achieving much better predictions on the LDO task than on the LCLO task. The synergy between two drugs when targeting a particular cell line changes less when you change the cell line then when you change one of the drugs.

On the LDCO task, SynergyIF performs better than DeepSynergy, CoSTCo, CP-WOPT and DSCP on most metrics across both compound screens. SynergyIF achieves similar performance as PRODeepSyn. The comparison between DTF and SynergyIF, though mostly statistically significant, yields mixed results, with DTF outperforming SynergyIF on some metrics and vice versa on others.

SynergyIF's hyperparameter search yielded unexpected results. The embedding dimension was consistently selected to be remarkably high (higher than the number of drugs or cell lines in the dataset). The rationale behind SynergyIF's utilization of IF is to first compress the two data modalities down to a low dimension, before fusing them together. It appears that this may not have been important. In future work, it would be beneficial to explore the use of Early Fusion as these results seem to indicate that it would achieve comparable results to IF while having significantly less complexity and training time.

In addition to the higher-than-expected embedding dimension, SynergyIF's hyperparameter search consistently found the weight on the reconstruction of the auxiliary data to be unimportant to the performance of SynergyIF as a whole. This at least implies that the reconstruction of the auxiliary data was unnecessary for SynergyIF's generation of high-quality predictions. It may also indicate that the auxiliary data itself was not particularly useful. This second claim will be investigated further in Chapter 4.

3.6 Summary

In conclusion, we developed SynergyIF, an algorithm that uses IF to combine tensor-based features with the physicochemical and genomic features of drugs and cell lines for ACDSP. SynergyIF generally outperformed DSCP and did not improve upon the SOTA performance. We discovered preliminary evidence that IF may not be well suited to the ACDSP task and that reconstruction of auxiliary data does not improve upon the predictive performance of SynergyIF. Finally, we show that SynergyIF learns to predict the average drug synergy across drug combinations in the training set on the Leave-Cell-Line-Out task.

Chapter 4 Physicochemical, Molecular Graph, Multiomic and Tensor-Based Features Offer Little Increase in Predictive Performance over Randomized Features for Anticancer Drug Synergy Prediction

4.1 Introduction

When no multi-way relations are present in our data, DL algorithms require high-quality features to enable them to make reasonable predictions. Intuitively, replacing these features with a unique identifier would eliminate the DL algorithm's ability to generalize to unseen data points. This intuition is challenged when multi-way relations are present in the data. It is possible that unique identifiers can make effective features in these cases as the DL algorithm may be able to leverage interactions that an entity has had with other entities to predict previously unseen relations. The Average algorithm introduced in section 2.3.3, performs a similar style of reasoning. On the LCLO problem, the Average algorithm takes the average synergy of a drug pair across cell lines in the training dataset and uses that average as a prediction for cell lines in the testing dataset. Though Average is not a DL algorithm, it is theoretically possible for an MLP to learn to make the same predictions as Average using only unique identifiers for the drugs and cell lines as input features.

The fact that SynergyIF learned to predict the average on the LCLO problem suggests that SynergyIF was doing precisely this. It is possible that SynergyIF was treating the physicochemical and genomic features of the drugs and cell lines as unique identifiers, ignoring their chemical and biological significance. SynergyIF did not significantly improve upon the

SOTA performance on the LDCO task, and the reconstruction of the auxiliary data was not found to be useful in the hyperparameter search. One possible explanation for this is that the addition of auxiliary data to tensor-based features offers no additional benefits over using one data source on its own. This may be the case if ACDSP algorithms are treating their input data as unique identifiers. Each individual data source works sufficiently well as a unique identifier.

In this chapter, we will investigate the hypothesis that SOTA ACDSP algorithms, including DSCP and SynergyIF, fail to learn meaningful patterns from their input data, instead treating it as a unique identifier. We will accomplish this by modifying the input data of these algorithms in such a way that makes it impossible for the algorithms discover meaningful patterns, such as by replacing the data with random numbers, and comparing the resulting performance with the algorithm trained on the original data. Finally, we will analyze the performance of an ACDSP algorithm using unique identifiers as features for the drugs and cell lines in the datasets.

4.2 Materials and Methods

To determine if ACDSP algorithms use physicochemical, molecular graph or tensor-based features of the drug and multiomic features of the cell line, we will randomize their input data and evaluate the effect this has on the predictive power of the algorithm. Additionally, we will analyze the effect that replacing the input data with a unique identifier has. If the algorithms depend on their input data to make high-quality predictions, we expect to see a significant decrease in the prediction performance of the algorithms that have randomized or unique identifier input data. If they do not, we would expect to see the prediction performance stay largely the same.

We chose different randomization strategies based on what best lets us randomize the input data without making any changes to the structure of the algorithm. This allows us to isolate the effects that the use of a particular data source has on the performance of an algorithm. When the input data to an ACDSP algorithm is tabular, we simply replace each entry of the drug and cell line feature matrices with random numbers drawn for a normal distribution. This is depicted in **Figure 4.1**. It is impossible for the algorithm to learn from the structure of the drug or cell line under this setting. We utilize this strategy for DeepSynergy.

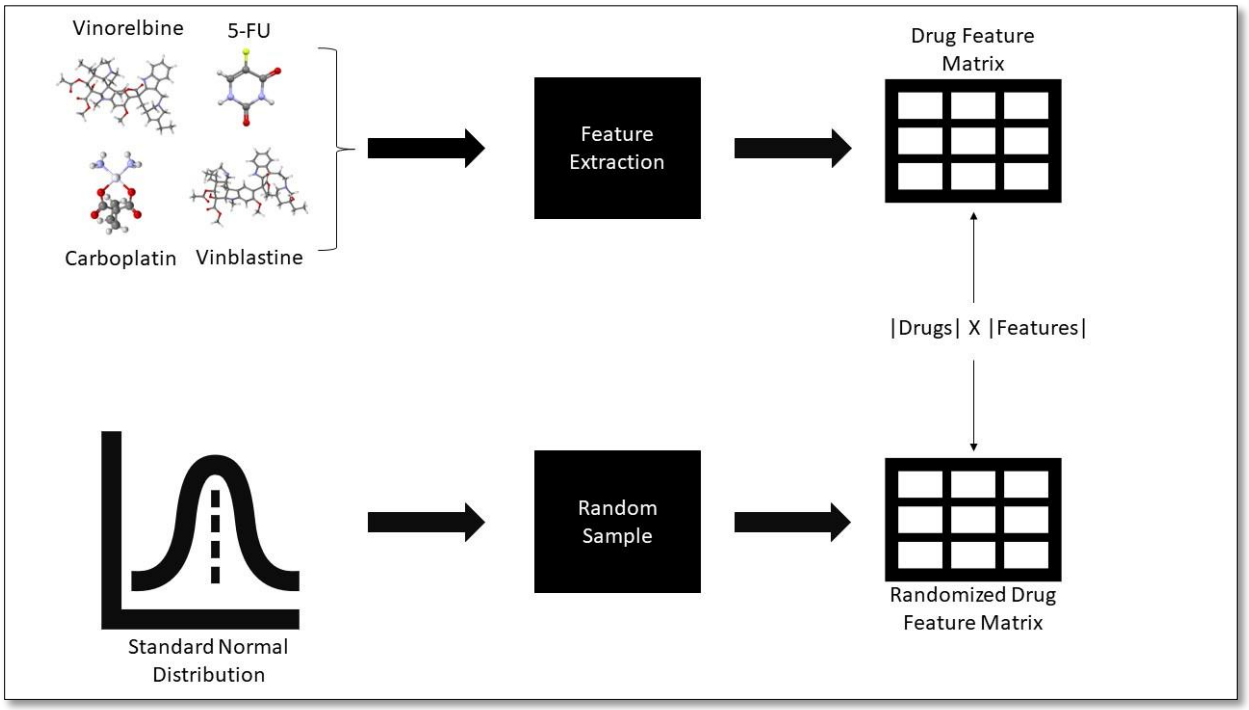

Figure 4.1 A depiction of the randomization strategy used in this chapter.

We also wish to investigate the effects of replacing the input data of ACDSP algorithms with unique identifiers. When the input data is non-tabular, this becomes quite complicated. For this reason, we choose to analyze DeepSynergy, as its input data is fully tabular. We replace its input data with unique identifiers by replacing its feature matrices with identity matrices.

Many SOTA algorithms use more complex data sources than can be stored in a table. For these algorithms, we are unable to apply the aforementioned randomization strategy. Instead of replacing feature vectors with random numbers, we permute the sequence of the drug and cell line features. In this way, the algorithm is forced to use features for a drug that correspond to the chemical structure of a different drug entirely. Likewise, the algorithm is forced to use features for a cell line that correspond to the multiomic structure of a different cell line entirely. This is depicted in Figure 4.2. Any inference based on the chemical or genomic structure of the drugs and cell line is impossible in this setting. Note that we permute the order of the drugs and cell lines once at the beginning of each experiment. This permutation remains consistent throughout the experiment. We utilize this strategy for DTF, DSCP, and SynergyIF, as they use a tensor as input features, and PRODeepSyn, as it uses a protein-protein Interaction network as input features for the cell lines.

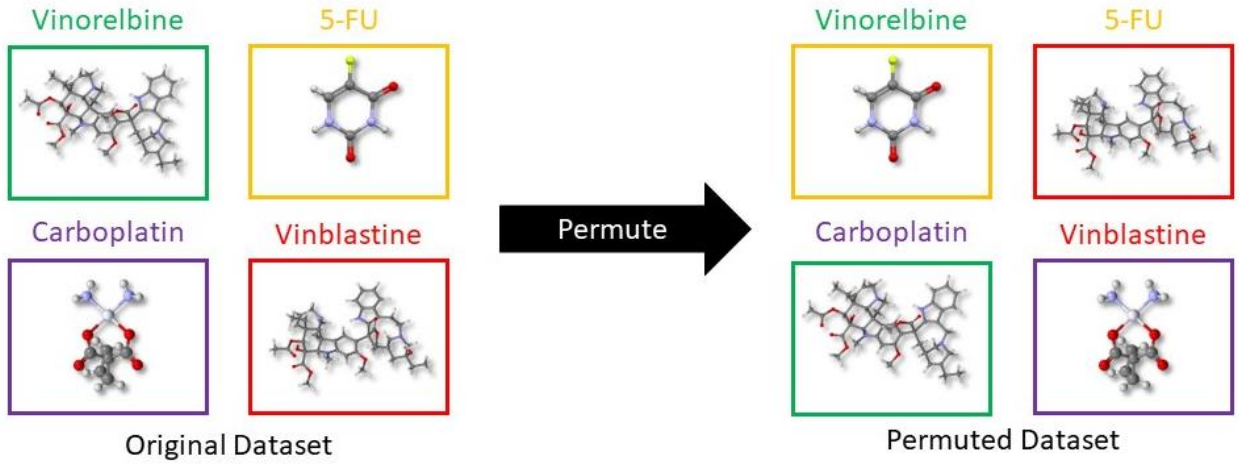

Figure 4.2 A depiction of the permutation strategy used in this chapter.

4.3 Data Sources and Algorithm Evaluation Strategy

Table 4.1 A summary of the algorithms, data sources, cross-validation strategies, metrics, and software considered in this chapter.

Algorithms
DTF [4], DeepSynergy [35], PRODeepSyn [37], DSCP, SynergyIF
Data Sources
Merick and Co [2], NCI-ALMANAC [47], DrugComb [65]
Cross-Validation Strategies
Leave-Drug-Combination-Out (LDCO), Leave-Cell-Line-Out (LCLO), Leave-Drug-Out (LDO)
Metrics
ROC-AUC, PR-AUC, Precision, Recall, F1-Score, BACC, Kappa, R2, MAE, MDAE, RMSE
Software
TensorFlow (2.4.1)[51], Tensorly (0.6.0) [63], SciPy (1.7.3) [64], Scikit-Learn (0.24.2) [58]

4.3.1 Drug Combination Compound Screens-

In addition to the Merick and Co and NCI-ALMANAC compound screens discussed in section 2.3.1.1, we utilize the DrugComb [65] compound screen. DrugComb contains 4150 drugs and 112 cell lines with 466’033 drug and cell line combinations. It is a combination of many

other compound screens, primarily Merick and Co and NCI-ALMANAC. We followed the preprocessing strategy outlined by Kuru et al. [66], after which the dataset contained 3040 drugs and 81 cell lines with 335’692 drug and cell line combinations. Each of these combinations were tested *in vitro* and the Loewe additivity score was recorded. Following previous literature [66] we consider combinations with a Loewe additivity score higher than 10 to be highly synergistic. As DrugComb is a large compound screen, it requires considerable computational resources to perform analysis on it. For this reason, only the LDCO cross-validation strategy is employed.

SynergyIF and DSCP scale cubically with the number of unique drugs and cell lines, making it computationally infeasible to run on a dataset with a large number of drugs and cell lines. PRODeepSyn’s protein-protein interaction network data was not readily available for the DrugComb dataset. For these reasons, these three algorithms were not evaluated on the DrugComb dataset.

4.3.2 Randomization Evaluation Strategy

We compare both a randomized and non-randomized version of many ACDSP algorithms on various datasets and cross-validation strategies. We report the Published performance of each algorithm by reporting the performance of the algorithm’s predictions after being trained using the data sources described in the algorithm’s original publication. We report the Randomized performance of each algorithm by reporting the performance of the algorithms’ predictions after being trained using randomized input data. We report the Difference by reporting the absolute difference between the Published and Randomized algorithms. We measure the Kappa score for the classification task and the MAE for the regression task and report the results in two separate experiments.

Additionally, we report the Significance by reporting the p-value resulting from McNemars's test [67] for significance and a Wilcoxon Signed-Rank test [68] for significance for the classification tasks and regression tasks, respectively. For McNemar's test, a p-value of less than 0.05 indicates that we have sufficient evidence to reject the null hypothesis that the Random and Published algorithms yield similar proportions of errors on the test set. Likewise, for the Wilcoxon Signed-Rank test, a p-value of less than 0.05 indicates that we have sufficient evidence to reject the null hypothesis that the difference between the absolute error of the predictions of the Randomized and Published algorithms on the test set are symmetric around 0 in favor of the alternative hypothesis that it is skewed to the left. In this case, the Published algorithm would have a lower mean absolute error over the entire test set. These conclusions are drawn at a 95% confidence level.

Each of the algorithms, both Published and Randomized, were trained 10 times on the train set, generating 10 predictions for each algorithm on the test set. To reduce the natural variation that arises during the training process of DL algorithms, we average these 10 predictions.

4.4 Results

4.4.1 Merick and Co Compound Screen

4.4.1.1 Leave-Drug-Combination-Out (LDCO)

On the LDCO regression task **(Table 4.3**), PRODeepSyn and SynergyIF were able to have their input data randomized without resulting in a significant difference in performance. The remaining algorithms showed significantly better performance when trained with

randomized data. On the LDCO classification task (**Table 4.2**), only SynergyIF performed worse when trained with randomized input data. DSCP and DeepSynergy were improved, and the remainder did not show a significant difference.

Table 4.2 A comparison of the classification performance of ACDSP algorithms with and without randomized input data on the Merick and Co compound screen using the LDCO cross-validation strategy. *(*) indicates statistical significance.*

Algorithm	Published Kappa	Randomized Kappa	Difference	Significance
DeepSynergy	0.46	0.47	0.02	0.03*
DTF	0.48	0.47	0.00	0.30
PRODeepSyn	0.46	0.48	0.02	0.12
DSCP	0.43	0.47	0.05	0.00*
SynergyIF	0.46	0.41	0.05	0.01*

Table 4.3 A comparison of the regression performance ACDSP algorithms with and without randomized input data on the Merick and Co compound screen using the LDCO cross-validation strategy. *(*) indicates statistical significance.*

Algorithm	Published MAE	Randomized MAE	Difference	Significance
DeepSynergy	10.18	9.61	0.58	1.00
DTF	9.91	9.67	0.24	1.00
PRODeepSyn	9.81	9.83	0.01	0.06
DSCP	10.25	9.93	0.32	1.00
SynergyIF	9.64	9.61	0.04	0.76

4.4.1.2 Leave-Drug-Out (LDO)

Randomizing DSCP's and SynergyIF's input data significantly improves performance on both the classification (**Table 4.4**) and regression (**Table 4.5**) task. Though randomizing DTF's

input data yields insignificant differences on the classification task, on the regression task, it yields worse predictions. For classification, the randomized version of PRODeepSyn performs worse. On the regression task the results are insignificant. Randomizing DeepSynergy's input data negatively affects classification performance and positively affects regression performance.

Table 4.4 A comparison of the classification performance of ACDSP algorithms with and without randomized input data on the Merick and Co compound screen using the LDO cross-validation strategy. *(*) indicates statistical significance.*

Algorithm	Published Kappa	Randomized Kappa	Difference	Significance
DeepSynergy	0.30	0.20	0.09	0.00*
DTF	0.21	0.24	0.03	0.08
PRODeepSyn	0.30	0.28	0.03	0.00*
DSCP	0.16	0.23	0.06	0.01*
SynergyIF	0.23	0.28	0.05	0.00*

Table 4.5 A comparison of the regression performance ACDSP algorithms with and without randomized input data on the Merick and Co compound screen using the LDO cross-validation strategy. *(*) indicates statistical significance.*

Algorithm	Published MAE	Randomized MAE	Difference	Significance
DeepSynergy	14.69	14.55	0.14	1.00
DTF	14.52	14.67	0.15	0.00*
PRODeepSyn	14.57	14.52	0.05	0.90
DSCP	15.53	15.12	0.41	1.00
SynergyIF	14.88	14.75	0.10	1.00

4.4.1.3 Leave-Cell-Line-Out (LCLO)

No algorithm performs worse when their input data has been randomized on both the regression (**Table 4.7**) and classification **(Table 4.6**) task. Only DeepSynergy's performance was improved by randomizing its input data. The remaining algorithms produce mixed and insignificant results. PRODeepSyn's randomized variant performs better on the regression task and worse on the classification task. DTF's performs worse on the regression task and better on the classification task. DSCP and SynergyIF's randomized variant perform better on the classification task with no significant difference on the regression task.

Table 4.6 A comparison of the classification performance of ACDSP algorithms with and without randomized input data on the Merick and Co compound screen using the LCLO cross-validation strategy. *(*) indicates statistical significance.*

Algorithm	Published Kappa	Randomized Kappa	Difference	Significance
DeepSynergy	0.30	0.41	0.11	0.00*
DTF	0.39	0.41	0.02	0.00*
PRODeepSyn	0.38	0.37	0.01	0.00*
DSCP	0.27	0.30	0.02	0.00*
SynergyIF	0.37	0.40	0.03	0.00*

Table 4.7 A comparison of the regression performance of ACDSP algorithms with and without randomized input data on the Merick and Co compound screen using the LCLO cross-validation strategy. *(*) indicates statistical significance.*

Algorithm	Published MAE	Randomized MAE	Difference	Significance
DeepSynergy	11.45	11.05	0.4	1.00
DTF	11.12	11.27	0.15	0.00*
PRODeepSyn	11.17	10.9	0.27	1.00
DSCP	10.91	10.94	0.03	0.07
SynergyIF	11.22	11.18	0.04	0.45

4.4.2 NCI-ALMANAC Compound Screen

4.4.2.1 Leave-Drug-Combination-Out (LDCO)

SynergyIF performed worse on both the regression (**Table 4.9**) and classification **(Table 4.8**) task when its input data was randomized. The remaining algorithms had mixed and insignificant results. The randomized variants of DTF and DSCP performed better on the classification task and worse on the regression task. DeepSynergy performed worse on the classification task and better on the regression task when its input data was replaced with random numbers. Though PRODeepSyn performed worse on the classification task with randomized data, its regression performance did not differ significantly.

Table 4.8 A comparison of the classification performance of ACDSP algorithms with and without randomized input data on the NCI-ALMANAC compound screen using the LDCO cross-validation strategy. *(*) indicates statistical significance.*

Algorithm	Published Kappa	Randomized Kappa	Difference	Significance
DeepSynergy	0.43	0.23	0.20	0.00*
DTF	0.46	0.47	0.01	0.00*
PRODeepSyn	0.42	0.36	0.06	0.00*
DSCP	0.40	0.44	0.03	0.00*
SynergyIF	0.47	0.46	0.01	0.00*

Table 4.9 A comparison of the regression performance ACDSP algorithms with and without randomized input data on the NCI-ALMANAC compound screen using the LDCO cross-validation strategy. *(*) indicates statistical significance.*

Algorithm	Published MAE	Randomized MAE	Difference	Significance
DeepSynergy	3.70	3.71	0.01	1.00
DTF	3.60	3.63	0.03	0.00*
PRODeepSyn	3.67	3.68	0.01	0.67
DSCP	3.67	3.76	0.10	0.00*
SynergyIF	3.63	3.68	0.05	0.00*

4.4.3 DrugComb Compound Screen

4.4.3.1 Leave-Drug-Combination-Out (LDCO)

Both DeepSynergy and DTF were significantly improved by the randomization of their input data on both the classification (**Table 4.10**) and regression (**Table 4.11**) task. DTF's improvement is small, though still significant, while DeepSynergy experienced a sizable increase in performance on both tasks.

Table 4.10 A comparison of the classification performance of ACDSP algorithms with and without randomized input data on the DrugComb compound screen using the LDCO cross-validation strategy. *(*) indicates statistical significance.*

Algorithm	Published Kappa	Randomized Kappa	Difference	Significance
DeepSynergy	0.42	0.45	0.02	0.00*
DTF	0.35	0.35	0.01	0.00*

Table 4.11 A comparison of the regression performance of ACDSP algorithms with and without randomized input data on the DrugComb compound screen using the LDCO cross-validation strategy. *(*) indicates statistical significance.*

Algorithm	Published MAE	Randomized MAE	Difference	Significance
DeepSynergy	6.20	5.76	0.44	1.00
DTF	5.71	5.69	0.02	1.00

4.4.4 Unique Identifier Experiment

When we compare DeepSynergy to its identity variant on the regression (**Table 4.12**) task, we mostly observe that the identity variant experiences either an improved MAE or insignificant results. On the Merick-LDO, Merick-LDCO and DrugComb-LDCO datasets, DeepSynergy obtains a significantly better MAE by replacing its input data with identity matrices. Only on NCI-ALMANAC-LDCO do we see a significantly worse MAE.

Table 4.12 A comparison of the regression performance of DeepSynergy with and without identity matrix input data, on every dataset and cross-validation strategy. *(*) indicates statistical significance.*

Dataset	Cross-Validation	Published MAE	Identity MAE	Difference	Significance
Merick	**LDCO**	10.18	9.99	0.19	0.23
	LDO	14.69	14.51	0.19	1.00
	LCLO	11.45	11.03	0.42	1.00
NCI-ALMANAC	**LDCO**	3.70	3.75	0.05	0.00*
DrugComb	**LDCO**	6.20	5.76	0.44	1.00

The effect is less clear when examining the classification (**Table 4.13**) task. On both the NCI-ALMANAC-LDCO and DrugComb-LDCO datasets, the use of identity matrices results in a worse Kappa score. On both the Merick-LDCO and Merick-LDO datasets the effect is not statistically significant. Only on the Merick-LCLO dataset do we observe a superior Kappa score by using identity matrices.

Table 4.13 A comparison of the classification performance of DeepSynergy with and without identity matrix input data, on every dataset and cross-validation strategy*. (*) indicates statistical significance.*

Dataset	Cross-Validation	Published Kappa	Identity Kappa	Difference	Significance
Merick	**LDCO**	0.46	0.45	0.00	0.19
	LDO	0.30	0.26	0.03	0.32
	LCLO	0.30	0.39	0.09	0.00*
NCI-ALMANAC	**LDCO**	0.43	0.39	0.04	0.00*
DrugComb	**LDCO**	0.42	0.39	0.03	0.00*

4.5 Discussion

Conventional intuition would strongly suggest that randomizing an algorithm's input data would catastrophically reduce its performance across nearly every metric, evaluation strategy and dataset. It is remarkable that, for ACDSP, doing so caused such mixed results. Some metrics improve with randomization, others get worse, and many more do not change significantly. These results are highly prone to changing with a change in dataset or evaluation strategy. Even more remarkably, though in some specific settings randomizing the data appears to hurt performance, the reduction is quite small. We would expect to see a much larger reduction by randomizing an algorithm's input data. Overall, it is not at all clear that the drug and cell line features, including ones based on TD, used by the ACDSP community actually aid in the predictive powers of the algorithms that use them. We provide support for the hypothesis that, by treating their input data as unique identifiers for the drugs and cell lines, the ACDSP algorithms

are learning to perform a style of reasoning similar to that performed by the Average baseline algorithm.

Since DeepSynergy was proposed in 2016, the approach most commonly used to improve upon DL-based ACDSP has been to extract drug and cell line features in progressively more sophisticated manners. When improvements have been observed in the resulting algorithms, they are typically attributed to this more sophisticated feature extraction. The evidence in this chapter challenges many of these conclusions. What follows are several examples of this.

In DeepSynergy's original publication [35], Preuer et al. say...

> "We have developed a novel Deep Learning based method, DeepSynergy, that predicts synergy scores of drug combinations for cancer cell lines with high accuracy. *The method requires cancer cell lines to be described by their genomic profiles, and the compounds to be represented by their chemical descriptor*" (emphasis is our own)

In this study, we have shown that instead of requiring genomic profiles and chemical descriptors, using random numbers often improves the performance of DeepSynergy. We show that what is actually required is that DeepSynergy uses input features that are unique from one another.

In PRODeepSyn's original publication [37], Wang et al. say...

> "Theoretically, PRODeepSyn integrates genomics data, transcriptomics data and the relationship between proteins into the final cell line embeddings, benefiting from the powerful ability of GCN to extract the structural features of the PPI network. The embeddings with multi-level information have been proved to be significantly helpful to predict the synergy scores of drug combinations in our experiments."

While PRODeepSyn certainly showed improved performance over most other algorithms both in their original publication as well as in many of the experiments in this thesis, the authors conclusion that this improvement is due to the multiomics and protein-protein interaction data is challenged by the fact that this data could be permuted without significantly hurting the regression performance of PRODeepSyn. It remains to discover what exactly causes PRODeepSyn improvement in performance. It is possible that the complexity that arises by adding additional data introduces an implicit regularization effect.

The rationale behind DTF is that it extracts features about each drug and cell line using TD on a drug synergy tensor. Our study suggests that the important aspect of these tensor features is that they are unique from one another, not that they describe how the drugs and cell lines behaves in the compound screen.

Though SCP has the ability to selectively extract important information from a tensor, it is not clear that it has any advantage over CP-WOPT at generating feature embeddings that are unique from one another. If it is indeed true that what matters is uniqueness, then it becomes clear as to why DSCP offers no clear advantage over DTF. In light of the fact that both DSCP, which uses tensor-based features, and DeepSynergy, which uses genomic and physicochemical-based features, can have their input data randomized with little effect on the performance of the algorithms, the failure of SynergyIF to achieve significantly better performance then DSCP and DeepSynergy is clear. SynergyIF combines both DSCP's and DeepSynergy's input data. Though it does this in a sophisticated manner, the combination of two sources of data that are not provably better than random is not likely to result in a data source that is better than random.

It is not clear from this analysis why ACDSP algorithms seem to struggle to learn from the patterns present in the physicochemical, genomic, multiomic and tensor-based features for

the drugs and cell lines. We postulate that it is due to small number of drugs and cell lines in the compound screens used by the research community. It is often assumed that, because of the large number of drug-drug-cell line triplets present in these compound screens, DL will have sufficient data to learn meaningful patterns. However, because the DL algorithms are required to learn patterns across drugs and cell lines features, we believe that the number of these play a larger role enabling the utility of drug and cell line features then the number of drug-drug-cell line triplets do. Based on this, we would expect to see the difference between randomized and non-randomized algorithms to be greater on the DrugComb dataset. We do not observe this in this analysis. However, DrugComb primarily consists of a combination of the NCI-ALMANAC and Merick and Co compound screens. For this reason, we may not expect to see results on DrugComb that differ from the results observed on NCI-ALMANAC and Merick and Co.

4.6 Summary

In conclusion, we discovered evidence that the multi-way structure of the anticancer drug synergy compound screens allows DL algorithms to make high quality predictions using unique identifiers as input data. We discovered evidence suggesting that ACDSP algorithms can have their input data randomized without significantly affecting their performance. We provide evidence that many of the assumptions made by past researchers regarding the reason for improvements shown by their algorithms were incorrect. In particular, the input data used is far less important than was previously believed by the research community at large. We show, instead, that the uniqueness of the data used appears to be more important than the source of the data used. Finally, we provide a reasoning for the lack luster performance of DSCP and SynergyIF. We strongly suggest that researchers include in their ablation studies a comparison

between a randomized and non-randomized version of their algorithms, as doing so will allow the readers to assess the utility of the data used.

Chapter 5 Conclusion and Future Work

5.1 Conclusion

In conclusion, we found that many of the drug and cell line features used by researchers for the task of ACDSP are not clearly better than randomized features. This challenges many of the assumptions made by ACDSP researchers regarding what has been the source of recent algorithm's predictive performance improvements. Additionally, we discovered evidence that SOTA ACDSP algorithms treat their input features as unique identifiers, ignoring their real-world significance.

We developed DSCP, a DTC algorithm that is trainable in an end-to-end fashion. We based DSCP off a TD algorithm, developed in this thesis, called SCP. We showed that DSCP made higher quality predictions on a synthetic tensor classification task, while using less resources than a two-step trainable version of DSCP. On the ACDSP task, we found that DSCP did not greatly improve upon the two-step trainable version and that this was likely because DSCP offers no advantage at generating unique identifiers for the drugs and cell lines.

To the best of our knowledge, we were the first to apply IF to tensor-based, physicochemical, and genomic features to design an ACDSP algorithm named SynergyIF. We found that IF did not clearly improve upon SOTA ACDSP algorithms and postulate that this is due to none of the data sources being clearly better than randomized features. Thus, the combination of them, no matter how sophisticated, is also not clearly better than randomized features.

5.2 Future Work

If it is indeed true that even the most sophisticated input features of SOTA ACDSP algorithms can be randomized without having a clear negative effect on the predictive performance of the algorithm, future work in the direction of developing ever more advanced features to be used by ACDSP algorithms has little hope of dramatically improving their performance. Additionally, changes to the DL component of these algorithms often are premised on utilizing underlying assumptions about the input data. It is our opinion that this direction will also bear little fruit, as it is likely the input features will simply be treated as unique identifiers. Thus, we believe it is of critical importance to investigate further and potentially circumvent the random/unique identifier input feature problem.

First, we believe it is important to confirm the existence of this phenomenon. We believe that repeating our experiments on two separate machines, one with non-random input features stored on it and one without. This will confirm that no data leakage is occurring. Additionally, independent validation of the results discovered in Chapter 4 of this thesis would be hugely beneficial to the research community. Finally, including more algorithms in our analysis would make it more comprehensive.

Second, we believe that obtaining a deep understanding of why this random/unique identifier issue occurs will be crucial for developing algorithms that do not suffer from it. We postulated that the cause was the small number of drugs and cell lines present in anticancer drug synergy compound screens, however, there is currently little evidence for or against this. Thus, designing a synthetic dataset that mimics the structure of anticancer drug synergy compound screens, but allows us to vary the number of unique drugs and cell lines would be immensely informative.

If the cause of the random/unique identifier is confirmed to result from the small number of drugs and cell lines, a future area of work would be to effectively utilize monotherapy drug compound screens. These compound screens contain far more drugs than drug synergy screens do. Thus, transfer learning [69] from monotherapy anticancer drug efficacy prediction to ACDSP may yield an effective solution to this problem. Another way to effectively utilize drug monotherapy compound screens would be to treat the problem of ACDSP as a semi-supervised [70] task with the monotherapy efficacy acting as weak labels.

Bibliography

[1] R. B. Mokhtari *et al.*, "Combination therapy in combating cancer," *Oncotarget*, vol. 8, no. 23, p. 38022, 2017, doi: 10.18632/ONCOTARGET.16723.

[2] J. O'Neil *et al.*, "An Unbiased Oncology Compound Screen to Identify Novel Combination Strategies," *Molecular Cancer Therapeutics*, vol. 15, no. 6, pp. 1155–1162, Jun. 2016, doi: 10.1158/1535-7163.MCT-15-0843.

[3] K. Fan, L. Cheng, and L. Li, "Artificial intelligence and machine learning methods in predicting anti-cancer drug combination effects," *Briefings in Bioinformatics*, vol. 22, no. 6, pp. 1–12, Nov. 2021, doi: 10.1093/BIB/BBAB271.

[4] Z. Sun, S. Huang, P. Jiang, and P. Hu, "DTF: Deep Tensor Factorization for predicting anticancer drug synergy," *Bioinformatics*, vol. 36, no. 16, pp. 4483–4489, 2020, doi: 10.1093/bioinformatics/btaa287.

[5] Q. Song, H. Ge, J. Caverlee, and X. Hu, "Tensor Completion Algorithms in Big Data Analytics," *ACM Transactions on Knowledge Discovery from Data*, vol. 13, no. 1, Nov. 2017, doi: 10.1145/3278607.

[6] H. Chen and J. Li, "DrugCom: Synergistic Discovery of Drug Combinations Using Tensor Decomposition," *Proceedings - IEEE International Conference on Data Mining, ICDM*, vol. 2018-Novem, pp. 899–904, Dec. 2018, doi: 10.1109/ICDM.2018.00108.

[7] F. Huang, X. Yue, Z. Xiong, Z. Yu, S. Liu, and W. Zhang, “Tensor decomposition with relational constraints for predicting multiple types of microRNA-disease associations,” *Briefings in Bioinformatics*, vol. 22, no. 3, pp. 1–11, 2021, doi: 10.1093/bib/bbaa140.

[8] M. Iwata *et al.*, “Predicting drug-induced transcriptome responses of a wide range of human cell lines by a novel tensor-train decomposition algorithm,” *Bioinformatics*, vol. 35, no. 14, pp. i191–i199, 2019, doi: 10.1093/bioinformatics/btz313.

[9] Y. Liu, J. Luo, and H. Wu, “miRNA-Disease Associations Prediction Based on Neural Tensor Decomposition,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12838 LNAI, pp. 312–323, Aug. 2021, doi: 10.1007/978-3-030-84532-2_28.

[10] T. G. Kolda and B. W. Bader, “Tensor Decompositions and Applications,” *Siam Review*, vol. 51, no. 3, pp. 455–500, Aug. 2009, doi: 10.1137/07070111x

[11] J. D. Carroll and J. J. Chang, “Analysis of individual differences in multidimensional scaling via an n-way generalization of ‘Eckart-Young’ decomposition,” *Psychometrika 1970 35:3*, vol. 35, no. 3, pp. 283–319, Sep. 1970, doi: 10.1007/BF02310791.

[12] L. R. Tucker, “Some mathematical notes on three-mode factor analysis,” *Psychometrika*, vol. 31, no. 3, pp. 279–311, Sep. 1966, doi: 10.1007/BF02289464.

[13] E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Mørup, “Scalable tensor factorizations for incomplete data,” *Chemometrics and Intelligent Laboratory Systems*, vol. 106, no. 1, pp. 41–56, Mar. 2011, doi: 10.1016/j.chemolab.2010.08.004.

[14] M. Filipović and A. Jukić, “Tucker factorization with missing data with application to low-n-rank tensor completion,” *Multidimensional Systems and Signal Processing*, vol. 26, no. 3, pp. 677–692, Jul. 2015, doi: 10.1007/S11045-013-0269-9.

[15] L. Yuan, Q. Zhao, L. Gui, and J. Cao, “High-order tensor completion via gradient-based optimization under tensor train format,” *Signal Processing-image Communication*, vol. 73, pp. 53–61, Apr. 2019, doi: 10.1016/j.image.2018.11.012.

[16] I. V. Oseledets, “Tensor-Train decomposition,” *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2295–2317, Jan. 2011, doi: 10.1137/090752286.

[17] D. Hong, T. G. Kolda, and J. A. Duersch, “Generalized Canonical Polyadic tensor decomposition,” *Siam Review*, vol. 62, no. 1, pp. 133–163, Jan. 2020, doi: 10.1137/18m1203626.

[18] H. Liu, M. Tsang, Y. Li, and Y. Liu, “COSTCO: A neural tensor completion model for sparse tensors,” *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 324–334, 2019, doi: 10.1145/3292500.3330881.

[19] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature 2015 521:7553*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/NATURE14539.

[20] A. Vaswani *et al.*, “Attention is All You Need,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, in NIPS’17. Red Hook, NY, USA: Curran Associates Inc., 2017, pp. 6000–6010.

[21] S. Fang, Z. Wang, Z. Pan, J. Liu, and S. Zhe, “Streaming Probabilistic Deep Tensor Factorization,” 2020, [Online]. Available: http://arxiv.org/abs/2007.07367

[22] H. Ishwaran and J. S. Rao, “Spike and slab variable selection: Frequentist and Bayesian strategies,” *Annals of Statistics*, vol. 33, no. 2, pp. 730–773, Apr. 2005, doi: 10.1214/009053604000001147.

[23] X. Wu, B. Shi, Y. Dong, C. Huang, and N. V. Chawla, “Neural tensor factorization for temporal interaction learning,” *WSDM 2019 - Proceedings of the 12th ACM International Conference on Web Search and Data Mining*, vol. 19, pp. 537–545, Jan. 2019, doi: 10.1145/3289600.3290998.

[24] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/NECO.1997.9.8.1735.

[25] A. Uddin, D. Zhou, X. Tao, C.-C. Chou, and D. Yu, “MLCTR: A Fast Scalable Coupled Tensor Completion Based on Multi-Layer Non-Linear Matrix Factorization,” 2021, [Online]. Available: http://arxiv.org/abs/2109.01773

[26] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 770–778, Dec. 2016, doi: 10.1109/CVPR.2016.90.

[27] B. Liu, L. He, Y. Li, S. Zhe, and Z. Xu, “NeuralCP: Bayesian Multiway Data Analysis with Neural Tensor Decomposition,” *Cognitive Computation*, vol. 10, no. 6, pp. 1051–1061, 2018, doi: 10.1007/s12559-018-9587-4.

[28] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” in *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, 2014.

[29] Y. Jin *et al.*, “Variational auto-encoder based Bayesian Poisson tensor factorization for sparse and imbalanced count data,” *Data Mining and Knowledge Discovery*, vol. 35, no. 2, pp. 505–532, Mar. 2021, doi: 10.1007/S10618-020-00723-7.

[30] S. Oh, S. Kim, R. A. Rossi, and S. Kumar, “Influence-guided Data Augmentation for Neural Tensor Completion,” *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, vol. 10, no. 21, 2021, doi: 10.1145/3459637.

[31] G. Pruthi, F. Liu, S. Kale, and M. Sundararajan, “Estimating Training Data Influence by Tracing Gradient Descent,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 19920–19930, 2020.

[32] J. Li, R. Zhao, J.-T. Huang, and Y. Gong, “Learning small-size DNN with output-distribution-based criteria,” *15th Annual Conference of the International Speech Communication Association*, Sep. 2014, doi: 10.21437/interspeech.2014-432.

[33] A. Narita *et al.*, “Tensor factorization using auxiliary information,” *Data Mining and Knowledge Discovery 2012 25:2*, vol. 25, no. 2, pp. 298–324, Jul. 2012, doi: 10.1007/S10618-012-0280-Z.

[34] E. Acar, T. G. Kolda, and D. M. Dunlavy, “All-at-once Optimization for Coupled Matrix and Tensor Factorizations,” May 2011, Accessed: Nov. 22, 2021. [Online]. Available: http://arxiv.org/abs/1105.3422

[35] K. Preuer, R. P. I. Lewis, S. Hochreiter, A. Bender, K. C. Bulusu, and G. Klambauer, "DeepSynergy: predicting anti-cancer drug synergy with Deep Learning," *Bioinformatics*, vol. 34, no. 9, pp. 1538–1546, May 2018, doi: 10.1093/BIOINFORMATICS/BTX806.

[36] T. Zhang, L. Zhang, P. R. O. Payne, and F. Li, "Synergistic Drug Combination Prediction by Integrating Multiomics Data in Deep Learning Models," *Methods in Molecular Biology*, vol. 2194, pp. 223–238, 2021, doi: 10.1007/978-1-0716-0849-4_12.

[37] X. Wang *et al.*, "PRODeepSyn: predicting anticancer synergistic drug combinations by embedding cell lines with protein–protein interaction network," *Briefings in Bioinformatics*, vol. 23, no. 2, pp. 1–14, Mar. 2022, doi: 10.1093/bib/bbab587.

[38] D. Szklarczyk *et al.*, "STRING v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets," *Nucleic Acids Research*, vol. 47, no. D1, pp. D607–D613, Jan. 2019, doi: 10.1093/NAR/GKY1131.

[39] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, Sep. 2016, Accessed: Jul. 17, 2023. [Online]. Available: https://arxiv.org/abs/1609.02907v4

[40] X. Liu, C. Song, S. Liu, M. Li, X. Zhou, and W. Zhang, "Multi-way relation-enhanced hypergraph representation learning for anti-cancer drug synergy prediction," *Bioinformatics*, Aug. 2022, doi: 10.1093/BIOINFORMATICS/BTAC579.

[41] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, “Hypergraph Neural Networks,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 3558–3565, Jul. 2019, doi: 10.1609/AAAI.V33I01.33013558.

[42] S. Bai, F. Zhang, and P. H. S. Torr, “Hypergraph convolution and hypergraph attention,” *Pattern Recognition*, vol. 110, p. 107637, Feb. 2021, doi: 10.1016/J.PATCOG.2020.107637.

[43] H. Julkunen *et al.*, “Leveraging multi-way interactions for systematic prediction of pre-clinical drug combination effects,” *Nature Communications 2020 11:1*, vol. 11, no. 1, pp. 1–11, Dec. 2020, doi: 10.1038/S41467-020-19950-Z.

[44] W. Zhang, Z. Xue, Z. Li, and H. Yin, “DCE-DForest: A Deep Forest Model for the Prediction of Anticancer Drug Combination Effects,” *Computational and Mathematical Methods in Medicine*, vol. 2022, pp. I93–I101, 2022, doi: 10.1155/2022/8693746.

[45] S. R. Stahlschmidt, B. Ulfenborg, and J. Synnergren, “Multimodal deep learning for biomedical data fusion: a review.,” *Briefings in Bioinformatics*, vol. 23, no. 2, pp. bbab569–bbab569, Mar. 2022, doi: 10.1093/BIB/BBAB569.

[46] G. V. Trunk, “A problem of dimensionality: a simple example,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 3, pp. 306–307, Jul. 1979, doi: 10.1109/tpami.1979.4766926.

[47] S. L. Holbeck *et al.*, “The National Cancer Institute ALMANAC: A Comprehensive Screening Resource for the Detection of Anticancer Drug Pairs with Enhanced

Therapeutic Activity," *Cancer Research*, vol. 77, no. 13, pp. 3564–3576, Jul. 2017, doi: 10.1158/0008-5472.CAN-17-0489.

[48] G. Golub, "Matrix computations," in *Johns Hopkins University Press eBooks*, 2013. doi: 10.56021/9781421407944.

[49] R. H. Byrd, L. Peihuang, and J. Nocedal, "A limited-memory algorithm for bound-constrained optimization," *SIAM Journal on Scientific Computing*, vol. 16, pp. 1190–1208, Mar. 1995, doi: 10.2172/204262.

[50] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, Dec. 2014, Accessed: Nov. 26, 2021. [Online]. Available: https://arxiv.org/abs/1412.6980v9

[51] M. Abadi *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems." 2015. [Online]. Available: https://www.tensorflow.org/

[52] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[53] J. Kossaifi, Y. Panagakis, A. Anandkumar, and M. Pantic, "TensorLy: Tensor Learning in Python," *Journal of Machine Learning Research*, vol. 20, no. 26, pp. 1–6, 2019, [Online]. Available: http://jmlr.org/papers/v20/18-277.html

[54] P. Virtanen *et al.*, “SciPy 1.0: fundamental algorithms for scientific computing in Python,” *Nature Methods*, vol. 17, no. 3, pp. 261–272, Mar. 2020, doi: 10.1038/S41592-019-0686-2.

[55] F. Pedregosa *et al.*, “SciKit-Learn: Machine Learning in Python,” *Journal of Machine Learning Research*, Oct. 2011, *12*(85), 2825–2830. http://jmlr.org/papers/v12/pedregosa11a.html

[56] S. LOEWE, “The problem of synergism and antagonism of combined drugs.,” *Arzneimittelforschung*, vol. 3, no. 6, pp. 285–290, Jun. 1953.

[57] D. S. Cao, Q. S. Xu, Q. N. Hu, and Y. Z. Liang, “ChemoPy: freely available python package for computational biology and chemoinformatics,” *Bioinformatics*, vol. 29, no. 8, pp. 1092–1094, Apr. 2013, doi: 10.1093/BIOINFORMATICS/BTT105.

[58] G. Landrum, “RDKit.” https://www.rdkit.org/

[59] A. Brazma *et al.*, “ArrayExpress--a public repository for microarray gene expression data at the EBI,” *Nucleic Acids Research*, vol. 31, no. 1, pp. 68–71, Jan. 2003, doi: 10.1093/NAR/GKG091.

[60] W. Talloen *et al.*, “I/NI-calls for the exclusion of non-informative genes: a highly effective filtering tool for microarray data,” *Bioinformatics*, vol. 23, no. 21, pp. 2897–2902, 2007, doi: 10.1093/BIOINFORMATICS/BTM478.

[61] W. J. Youden, “INDEX FOR RATING DIAGNOSTIC TESTS”, doi: 10.1002/1097-0142.

[62] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A Next-generation Hyperparameter Optimization Framework,” *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2623–2631, Jul. 2019, doi: 10.1145/3292500.3330701.

[63] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001, doi: 10.1023/A:1010933404324/METRICS.

[64] F. Hutter, H. Hoos, and K. Leyton-Brown, “An Efficient Approach for Assessing Hyperparameter Importance.” *Proceeding of Machine Learning Research*, pp. 754–762, Jan. 27, 2014. Accessed: Jul. 15, 2023. [Online]. Available: https://proceedings.mlr.press/v32/hutter14.html

[65] B. Zagidullin *et al.*, “DrugComb: an integrative cancer drug combination data portal,” *Nucleic Acids Research*, vol. 47, no. W1, pp. W43–W51, Jul. 2019, doi: 10.1093/NAR/GKZ337.

[66] H. I. Kuru, O. Tastan, and A. E. Cicek, “MatchMaker: A deep learning framework for drug synergy prediction,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 19, no. 4, pp. 2334–2344, Jul. 2022, doi: 10.1109/tcbb.2021.3086702.

[67] T. G. Dietterich, “Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms,” *Neural Computation*, vol. 10, no. 7, pp. 1895–1923, Oct. 1998, doi: 10.1162/089976698300017197.

[68] F. Wilcoxon, “Individual Comparisons by Ranking Methods,” *Biometrics Bulletin*, vol. 1, no. 6, p. 80, Dec. 1945, doi: 10.2307/3001968.

[69] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010, doi: 10.1109/tkde.2009.191

[70] D. P. Kingma, S. Mohamed, D. Jimenez Rezende, and M. Welling, “Semi-supervised Learning with Deep Generative Models,” *Advances in Neural Information Processing Systems*, vol. 27, 2014, Accessed: Jul. 12, 2023. [Online]. Available: http://arxiv.org/abs/1406.5298

Supplementary

Table S.1 The results of the hyperparameter search for CP-Wopt on the Merick and Co compound screen using the LDCO cross-validation strategy.

Hyperparameter	Considered Values	Chosen Value	Importance
Rank	1-512	183	90%
Batch Size	512, 1024, 2048	512	1%

Table S.2 The results of the hyperparameter search for DTF on the Merick and Co compound screen using the LDCO cross-validation strategy.

Hyperparameter	Considered Values	Chosen Value	Importance
MLP Architecture	[1024 1024 512], [2048 2048 1024], [2048 1024 512], [2048 2048 2048], [1024 1024 1024], [2048 2048], [1024 1024], [512 512], [1024 512]	[1024 512]	62%
Hidden Layers DR	0-0.5	0.2	8%
Input Layers DR	0-0.2	0.1	27%
Batch Size	512, 1024, 2048	512	3%

Table S.3 The results of the hyperparameter search for DeepSynergy on the Merick and Co compound screen using the LDCO cross-validation strategy.

Hyperparameter	Considered Values	Chosen Value	Importance
MLP Architecture	[8192 4096 2048], [4096 4096 4096], [4096 2048 1024], [2048 2048 2048], [8192 8192], [8192 4096], [4096 4096], [4096 2048], [2048 2048]	[4096 4096 4096]	28%
Hidden Layers DR	0-0.5	0.1	64%
Input Layers DR	0-0.2	0.1	2%
Batch Size	512, 1024, 2048	512	6%

Table S.4 The results of the hyperparameter search for CoSTCo on the Merick and Co compound screen using the LDCO cross-validation strategy.

Hyperparameter	Considered Values	Chosen Value	Importance
Rank	1-512	406	22%
Layer 1 DR	0-0.5	0.5	2%
Layer 2 DR	0-0.5	0.5	4%
Layer 3 DR	0-0.5	0.1	4%
Regularization Constant	0-10	0.42	65%
Batch Size	512, 1024, 2048	512	4%

Table S.5 The results of the hyperparameter search for PRODeepSyn on the Merick and Co compound screen using the LDCO cross-validation strategy.

Hyperparameter	Considered Values	Chosen Value	Importance
MLP Architecture	[8192 4096 2048], [4096 4096 4096], [4096 2048 1024], [2048 2048 2048], [8192 8192], [8192 4096], [4096 4096], [4096 2048], [2048 2048]	[4096 4096]	60%
Hidden Layers DR	0-0.5	0.5	4%
Input Layers DR	0-0.2	0	2%
Batch Size	512, 1024, 2048	512	34%

Table S.6 The results of the hyperparameter search for CP-Wopt on the Merick and Co compound screen using the LDO cross-validation strategy.

Hyperparameter	Considered Values	Chosen Value	Importance
Rank	1-512	34	38%
Batch Size	512, 1024, 2048	1024	62%

Table S.7 The results of the hyperparameter search for DTF on the Merick and Co compound screen using the LDO cross-validation strategy.

Hyperparameter	Considered Values	Chosen Value	Importance
MLP Architecture	[1024 1024 512], [2048 2048 1024], [2048 1024 512], [2048 2048 2048], [1024 1024 1024], [2048 2048], [1024 1024], [512 512], [1024 512]	[2048 1024 512]	23%
Hidden Layers DR	0-0.5	0.3	51%
Input Layers DR	0-0.2	0	25%
Batch Size	512, 1024, 2048	1024	0%

Table S.8 The results of the hyperparameter search for DeepSynergy on the Merick and Co compound screen using the LDO cross-validation strategy.

Hyperparameter	Considered Values	Chosen Value	Importance
MLP Architecture	[8192 4096 2048], [4096 4096 4096], [4096 2048 1024], [2048 2048 2048], [8192 8192], [8192 4096], [4096 4096], [4096 2048], [2048 2048]	[8192 4096]	35%
Hidden Layers DR	0-0.5	0.5	40%
Input Layers DR	0-0.2	0.1	2%
Batch Size	512, 1024, 2048	512	22%

Table S.9 The results of the hyperparameter search for CoSTCo on the Merick and Co compound screen using the LDO cross-validation strategy.

Hyperparameter	Considered Values	Chosen Value	Importance
Rank	1-512	211	7%
Layer 1 DR	0-0.5	0	3%
Layer 2 DR	0-0.5	0.5	42%
Layer 3 DR	0-0.5	0.2	8%
Regularization Constant	0-10	0.03	39%
Batch Size	512, 1024, 2048	512	0%

Table S.10 The results of the hyperparameter search for PRODeepSyn on the Merick and Co compound screen using the LDO cross-validation strategy.

Hyperparameter	Considered Values	Chosen Value	Importance
MLP Architecture	[8192 4096 2048], [4096 4096 4096], [4096 2048 1024], [2048 2048 2048], [8192 8192], [8192 4096], [4096 4096], [4096 2048], [2048 2048]	[8192 8192]	12%
Hidden Layers DR	0-0.5	0.4	10%
Input Layers DR	0-0.2	0	77%
Batch Size	512, 1024, 2048	1024	0%

Table S.11 The results of the hyperparameter search for CP-Wopt on the Merick and Co compound screen using the LCLO cross-validation strategy.

Hyperparameter	Considered Values	Chosen Value	Importance
Rank	1-512	129	91%
Batch Size	512, 1024, 2048	512	9%

Table S.12 The results of the hyperparameter search for DTF on the Merick and Co compound screen using the LCLO cross-validation strategy.

Hyperparameter	Considered Values	Chosen Value	Importance
MLP Architecture	[1024 1024 512], [2048 2048 1024], [2048 1024 512], [2048 2048 2048], [1024 1024 1024], [2048 2048], [1024 1024], [512 512], [1024 512]	[2048 2048 1024]	22%
Hidden Layers DR	0-0.5	0.4	37%
Input Layers DR	0-0.2	0.2	39%
Batch Size	512, 1024, 2048	1024	2%

Table S.13 The results of the hyperparameter search for DeepSynergy on the Merick and Co compound screen using the LCLO cross-validation strategy.

Hyperparameter	Considered Values	Chosen Value	Importance
MLP Architecture	[8192 4096 2048], [4096 4096 4096], [4096 2048 1024], [2048 2048 2048], [8192 8192], [8192 4096], [4096 4096], [4096 2048], [2048 2048]	[8192 4096 2048]	81%
Hidden Layers DR	0-0.5	0.5	17%
Input Layers DR	0-0.2	0	2%
Batch Size	512, 1024, 2048	512	0%

Table S.14 The results of the hyperparameter search for CoSTCo on the Merick and Co compound screen using the LCLO cross-validation strategy.

Hyperparameter	Considered Values	Chosen Value	Importance
Rank	1-512	159	41%
Layer 1 DR	0-0.5	0.3	4%
Layer 2 DR	0-0.5	0.3	9%
Layer 3 DR	0-0.5	0.2	11%
Regularization Constant	0-10	0	34%
Batch Size	512, 1024, 2048	512	1%

Table S.15 The results of the hyperparameter search for PRODeepSyn on the Merick and Co compound screen using the LCLO cross-validation strategy.

Hyperparameter	Considered Values	Chosen Value	Importance
MLP Architecture	[8192 4096 2048], [4096 4096 4096], [4096 2048 1024], [2048 2048 2048], [8192 8192], [8192 4096], [4096 4096], [4096 2048], [2048 2048]	[8192 4096]	60%
Hidden Layers DR	0-0.5	0.5	34%
Input Layers DR	0-0.2	0.1	4%
Batch Size	512, 1024, 2048	1048	3%

Table S.16 The results of the hyperparameter search for CP-Wopt on the NCI-ALMANAC compound screen using the LDCO cross-validation strategy.

Hyperparameter	Considered Values	Chosen Value	Importance
Rank	1-512	415	99%
Batch Size	512, 1024, 2048	1024	1%

Table S.17 The results of the hyperparameter search for DTF on the NCI-ALMANAC compound screen using the LDCO cross-validation strategy.

Hyperparameter	Considered Values	Chosen Value	Importance
MLP Architecture	[1024 1024 512], [2048 2048 1024], [2048 1024 512], [2048 2048 2048], [1024 1024 1024], [2048 2048], [1024 1024], [512 512], [1024 512]	[2048 2048 2048]	78%
Hidden Layers DR	0-0.5	0.1	11%
Input Layers DR	0-0.2	0.2	7%
Batch Size	512, 1024, 2048	1024	4%

Table S.18 The results of the hyperparameter search for DeepSynergy on the NCI-ALMANAC compound screen using the LDCO cross-validation strategy.

Hyperparameter	Considered Values	Chosen Value	Importance
MLP Architecture	[8192 4096 2048], [4096 4096 4096], [4096 2048 1024], [2048 2048 2048], [8192 8192], [8192 4096], [4096 4096], [4096 2048], [2048 2048]	[2048 2048 2048]	21%
Hidden Layers DR	0-0.5	0.4	74%
Input Layers DR	0-0.2	0.2	4%
Batch Size	512, 1024, 2048	1024	2%

Table S.19 The results of the hyperparameter search for CoSTCo on the NCI-ALMANAC compound screen using the LDCO cross-validation strategy.

Hyperparameter	Considered Values	Chosen Value	Importance
Rank	1-512	84	100%
Regularization Constant	0-10	0.76	0%
Batch Size	512, 1024, 2048	512	0%

Table S.20 The results of the hyperparameter search for PRODeepSyn on the NCI-ALMANAC compound screen using the LDCO cross-validation strategy.

Hyperparameter	Considered Values	Chosen Value	Importance
MLP Architecture	[8192 4096 2048], [4096 4096 4096], [4096 2048 1024], [2048 2048 2048], [8192 8192], [8192 4096], [4096 4096], [4096 2048], [2048 2048]	[2048 2048 2048]	85%
Hidden Layers DR	0-0.5	0.3	2%
Input Layers DR	0-0.2	0.2	11%
Batch Size	512, 1024, 2048	1024	1%